BEST AVAILABLE COPY

BU

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number: 0 067 667 A2

(12) # EUROPEAN PATENT APPLICATION

(21) Application number: 82302998.8

(22) Date of filing: 10.06.82

(51) Int. Cl.³: G 06 F 9/22

(30) Priority: 11.06.81 US 272507
11.06.81 US 272777
11.06.81 US 272541

(43) Date of publication of application: 22.12.82 Bulletin 82/51

(84) Designated Contracting States: AT BE CH DE FR GB IT LI LU NL SE

(71) Applicant: DATA GENERAL CORPORATION
Route 9
Westboro Massachusetts 01581(US)

(72) Inventor: Beauchamp, Robert William
16 Shadowbrook Lane
Milford Massachusetts 01757(US)

(72) Inventor: Blau, Jonathan Seth
135 Riviera Drive Apt. 341
Los Gatos California 95030(US)

(72) Inventor: Stein, James B.
25-10 Royal Crest Lane
Marlboro Massachusetts 01752(US)

(74) Representative: Pears, David Ashley et al,
REDDIE & GROSE 16 Theobalds Road
London WC1X 8PL(GB)

(54) **Data processing system.**

(57) The system employs a fixed point arithmetic unit (ALU) and a separate floating point arithmetic unit (FPU) controlled by a common microinstruction store. Most fields of the 80-bit microinstructions control both the ALU and the FPU, including a source field ALUS an operation field ALUOP. A further 2-bit field EXT determines whether the FPU performs double or single precision arithmetic, in which the operation defined by ALUOP is carried out respectively by all of 16 4-bit slice units and by the top 6 slice units only for 56 and 24 bit results, with 8 guard bits in either case. As well as a destination field ALUD for the ALU there is a separate destination field FPUD for the FPU and each such field is set to disable the output of its respective unit when the output is required from the other unit. Carry-in bits for bits 55 and 23 are entered at bit 63 and propagated through the lower bit slice units to avoid multiplexing in to the higher bit slice units.

EP 0 067 667 A2

Croydon Printing Company Ltd.

DATA PROCESSING SYSTEM

This invention relates to data processing systems and is concerned more particularly with the means used to perform arithmetic computations.

Data processing systems utilize arithmetic units for performing fixed or floating point calculations. One such system, for example, is utilized in a 32-bit processing system made and sold by Data General Corporation of Westboro, Massachusetts under the Registered Trade Mark ECLIPSE MV/8000. Such system utilizes an arithmetic logic unit (ALU) which can perform both fixed point calculations and floating point calculations.

In performing floating point calculations, the arithmetic logic unit of such system can be operated to produce either "single precision" accuracy wherein the floating point mantissa result comprises 24 bits (the remaining 8 bits being designated as "guard bits" for use in rounding operations) or "double precision" accuracy wherein the floating point mantissa result comprises 24 bits (the remaining 8 bits being designated as "guard bits" for use in rounding operations) or "double precision" accuracy in which the calculation is effectively a 64-bit operation wherein the floating point mantissa result comprises 56 bits (the remaining 8 bits being used for rounding in the same manner as in single precision operations). In the present ECLIPSE MV/8000 system, double precision operations are performed by the arithmetic logic unit as two sequential 32-bit operations so that the overall time required for double precision is substantially greater than that required for single precision operation.

It is desirable to be able to perform the double precision operation in substantially less time without greatly increasing the overall complexity or cost of the system in this regard.

An appropriate technique for doing so which has been suggested involves the performance of floating point operations by a floating point arithmetic unit which is separate from the arithmetic unit which performs fixed point operations. The floating point unit can be capable of performing its operation on 32-bit or 64-bit words to achieve both single and double precision operations. Thus, such a system would utilize both an arithmetic logic unit (ALU) and a

floating point unit (FPU), each having its own control store for storing its own microinstructions (i.e., microcode) which are used in performing the separate types of arithmetic calculations involved. Accordingly, the ALU control store would provide microcode for performing 32-bit fixed point operations while the FPU control store would provide microcode for performing both 32-bit and 64-bit floating point operations. The time required to perform a complete double precision floating point operation would be reduced over that required to perform such double precision floating point operations using only a single arithmetic logic unit which can operate only on 32-bit words at a time. Such an approach, however, requires the provision of an additional different control store with entirely different microcode for controlling the operation of the FPU independently and separately from the control store and microcode operation of the ALU.

Alternatively, it has been suggested that a single control store be used, which store, however, is considerably enlarged in comparison with that previously required when using a single arithmetic logic unit so as to provide completely different and separate sets of control fields for controlling the operation of the ALU and for controlling the operation of the FPU. The use of such an enlarged control store is required in order to accommodate a sufficient number of control fields to assure that such units are independently and effectively separately controlled.

It is desirable, however, to be able to provide the above desired operation without the need for two separate control stores or for such an enlarged control store. With a view to meeting this need, the present invention provides a data processing system as defined in claim 1 below.

Thus, in practising the invention, fixed point operations, as well as single and double precision floating point operations, can be achieved utilizing an ALU board and a separate FPU board, both of which are controlled through the use of a single, common control store having common control fields. In effect, then, control of both boards is achieved using substantially the same overall microcode, except for the microcode which relates to floating point operations. The general configuration of the FPU board is substantially similar to the general configuration of the

ALU board and both boards share common microinstruction fields for the control of their operations, the FPU board requiring the use of only two additional fields for controlling destination operations and for controlling single and double precision floating point operations. Such additional 3-bit and 2-bit fields, respectfully, can be readily accommodated in the single previously used control store without enlargement thereof. The remaining microcode fields are common to both ALU and FPU boards, the FPU board utilizing such microcode when a floating point operation is required (the ALU board being effectively disabled) and the ALU board using such microcode when non-floating point operation is required (the FPU board being effectively disabled).

A double precision operation can be performed at a speed which is substantially greater than that required when using the conventional 32-bit single ALU board operation mentioned above. Further, since substantially the same microcode is used for both ALU and FPU operations, it is not necessary to rewrite, or to add to, the previously used microcode for non-floating point operations except for minimal control field additions needed to distinguish between ALU or FPU operations, such operations otherwise using the same control fields as used for the previous ALU operation. Because the ALU board and the FPU board share the same fields, except for the added destination and single/double precision fields needed for the FPU, the complexity and costs of the system are reduced over systems which require two complete separate control stores or two completely separate sets of control fields of a single control store.

Moreover, the system is readily adapted to implementing an improved technique for providing floating point arithmetic rounding operations in either single or double precision modes, and the invention comprises also a data processing system as defined in claim 11.

The invention will be described in more detail, by way of example, with reference to the accompanying drawings, in which:

Figs. 1 and 2 make up a block diagram of an arithmetic logic unit as used in the prior ECLIPSE MV/8000 unit;

Fig. 3 and 4 make up a block diagram of a floating point unit in accordance with the invention; and

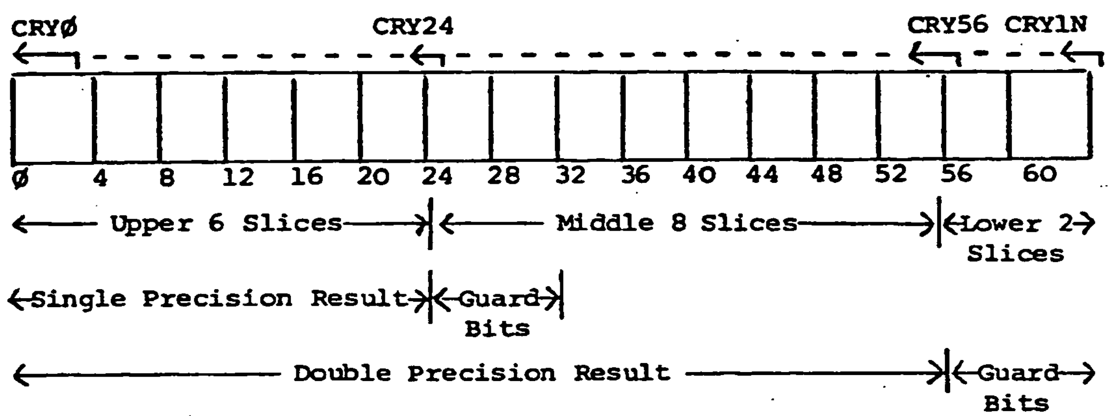
Figs. 5-36 depict specific logic diagrams for implementing the block diagram of Fig. 2 of the floating point unit of the invention.

At the end of this description there appear two tables, namely:

TABLE 1 shows a chart defining the control store mode (CSM) and extension (EXT) fields as used in floating point microinstructions in accordance with the invention; and

TABLE 2 shows a chart defining the comparable control store mode (CSM) fields of the previously used MV/8000 system.

When using a floating point unit for providing either single or double precision operation and calculating the mantissa value, the 64-bit word which results comprises the following portions:



During a single precision operation only 32 bits (bits 0-31) of the 64-bit words are used (bits 32-63 are "zero). The floating point mantissa result comprises bits 0-23 while bits 24-31 are designated as "guard" bits which are utilized in a rounding operation, as discussed in more detail below. When providing double precision operation the mantissa floating point result comprises bits 0-55, with bits 56-63 operating as the "guard" bits thereof.

In performing both fixed and floating point arithmetic operations in the aforesaid ECLIPSE MV/8000 system, a single arithmetic logic unit (ALU) as shown in broad block diagram form in Figs. 1 and 2 is used. The operation thereof is known in the art with respect to the currently available MV/8000 system. In accordance with the invention, a separate floating point unit (FPU) is used for performing floating point calculations, such unit having the broad block diagram configuration shown in Figs. 3 and 4. As

can be seen therein, the overall operational block diagram of the FPU is substantially similar to that of the previously utilized ALU.

Control of the operation of the ALU in the previous MV/8000 system is determined in accordance with specifically devised micro-code, a typical 80-bit microinstruction format as used therein, for example, being described below:

NAC	AREGS	BREGS	CSM	D1ST	D2ND	SHFT	ALUS	ALUOP	
20	4	4	4	2	4	4	3	3	
0	20	24	28	32	34	38	42	45	47

ALUD	CRYINS	RAND	LAC	CPDS	MEMS	MEMC	FPUD	EXT	UPAR
3	1	10	2	5	3	2	3	2	1
48	51	52	62	64	69	72	74	77	79

In this format bits 74-78 were unused in the MV/8000 system and the microcode field designated as the CSM (control store mode) controlled operations of the ALU for both fixed and floating point mode. In accordance with the invention, control of both the separate ALU and the FPU boards is achieved through the use of essentially the same microcode control store and microinstruction format wherein the five previously unused bits 74-78 are used to provide two additional fields for controlling the operation of the floating point unit destination registers (the FPUD field bits 74-76) and for controlling operation for either single or double precision results (the EXT field bits 77 and 78), as described in more detail below.

Thus, in the invention the microinstruction fields which control the operation of the floating point unit are the control store mode (CSM) field bits 28-31, the 3-bit bloating point unit destination FPUD) field bits 74-76, and the 2-bit extension (EXT) field, bits 77 and 78. As in the previous ECLIPSE MV/8000 unit, the operations for both the arithmetic logic unit and the floating point unit are cycled twice for each microinstruction (a first half-cycle and a second half-cycle) and the CSM, FPUD and EXT fields define the sources of control during each of the half-cycles.

A summary of the control store mode field and the extension field operations is shown in TABLE 1 for the various operations which are performed. As in the MV/8000 the floating point operation is performed in 4-bit slices, the overall 64-bit data words utilized

comprising sixteen 4-bit slices, the single precision floating point result comprising six 4-bit slices (the "upper" six slices) and the double precision 56-bit result comprising fourteen 4-bit slices (the "upper" six slices and the "middle" eight slices) and the guard bits for use in the double precision mode comprising two 4-bit slices (the "lower" two slices). In the first twelve mathematical operations performed, such operations can be either single or double precision operations as determined by the extension (EXT) field bits.

In the control store mode and extension fields described in TABLE 1, the FMATH, FFIXP, FGEN, FATU, BOUT, QDEC, QINC and QADD modes use both half-cycles 1 and 2 for their normal operations (e.g., A op B $\rightarrow$ B). The remaining modes perform two FPU operations. Most tests, and TLCH (nibble shift input) look at only first half data during these modes. The SMATH, SFIXP, SGEN and SATU modes perform an operation during the first half-cycle but utilize the data but (DBUS) during the second half-cycle. The NORM and PRESC modes are used to normalize and prescale the floating point operands, respectively, while the MPY mode implements a double cycle 2-bit Booth's multiply algorithm and the DIV mode implements a double cycle non-restoring divide operation, the latter operations both being conventional and well known in the art.

The first twelve mathematical modes can be used for either double or single precision accuracy and such operation is determined by the EXT field as shown in TABLE 1. Thus, for a double precision operation, where all sixteen 4-bit slices are utilized, the micro-order from the appropriate field of the specified microinstruction (uI) or the DBUS use controls all sixteen slices. This is designated in TABLE 1 by the presence of uI16 or the designations DZ16 and OR16. In the appropriate ALU and FPU fields, i.e., the ALUS (ALU source), the ALUOP (ALU operation field and the FPUD (FPU destination) field. The designation " ≠ " during the first half cycle for the FPUD field merely indicates that no FPUD clocking takes placed (i.e., there is no FPU destination oepration for such mode during the first half-cycle) and the designation "@16" indicates that the micro-order will defer to a pre-decoded or "forced" value.

For a single precision operation the lower ten 4-bit slices (bits 24-63) are zero since only bits 0-23 are used. In the particular operation of the floating point unit an "AND" operation

in which one of the operands is zero at each bit slice provides for a propagation of an input carry (CRY) bit through each bit slice to the output carry bit so that the FPU "CRYIN" is propagated throughout the lower ten 4-bit slices for single precision operation. The basis for such operation of the FPU computation circuit in this regard is described in more detail below with respect to the description of rounding techniques discussed at a later point herein. While zeros are present in the lower ten 4-bit slices the micro-order from the appropriate field of the specified microinstruction (uI) controls the upper six 4-bit slices. For example, in the first half-cycle for all of the four Fxxx modes shown in TABLE 1, single precision operation (EXT field 2) is designated by the legends "u16/DZ10" (indicating the operands of the upper six slices are under microinstruction control and the operands of the lower ten slices are the DBUS and zero) for the ALUS field and "u16/AND10" (indicating the operands of the upper six slices are under microinstruction control and each of the lower ten slices receive an AND operation) for the ALUOP field. The FPUD field is not used in the first half-cycle, while such field is under microinstruction control during the second half-cycle.

The Control Store Mode for controlling operation of the ALU utilizes substantially the same microcode, as shown by the CSM characteristics depicted in TABLE 2 (which is also the control store mode set up to control operation in the previously designed ECLIPSE MV/8000 system). As can be seen therein, where there is no separately defined single or double precision operation for the ALU, effectively the same CSM operations are performed as for the FPU operation and no EXT field is required. Thus, for ALU operation control field bits 74-78 of the above microinstruction are not utilized (no FPUD field and no EXT field controls are required). In each case, for both ALU and FPU operations, the ALUS and the ALUOP field values represent exactly the same operations as shown below.

The ALU is cycled twice for each micro-instruction. The CSM field defines the sources of control during the two half cycles.

The Fxxx modes use both cycles for normal A + B → B operation. The remaining modes perform 2 ALU operations. Most tests, LA, and TLCH (nibble shift input) look at only first half data during these modes. Sxxx modes are useful for setting up tests or memory starts

with the second half used for reading memory, O-DS, or the nibble shifter. BOUT is similar but allows memory write in the second half. The Qxxx modes allow two registers to be loaded in one cycle. The Q register is generally used to contain an address which will be incremented or decremented before each memory start. MPY and DIV implement double cycles of the multiply and divide algorithms.

ALUS FIELD (R,S)	
AQ	0
AB	1
ZQ	2
ZB	3
ZA	4
DA	5
DQ	6
DZ	7

ALUOP FIELD		
ADD	0	(R + S)
SUB	1	(S - R)
RSB	2	(R - S)
OR	3	(R or S)
AND	4	(R * S)
ANC	5	(R' *S)
XQR	6	(R xor S)
XNR	7	(R xor S)'

For the ALU operations the ALUD field and for FPU operations the FPUD field are also shown below.

ALUD FIELD

Mnem	Value	Description
NLD	0	No load; Y <0-31> = ALU <0-31>
QREG	1	Load QREG only; Y <0-31> = ALU <0-31>
BREG	2	Load BREG only; Y <0-31> = ALU <0-31>
AOUT	3	Load BREG only; Y <0-31> = AREG <0-31>
RSHB	4	Load BREG with ALU shifted right one bit; LINK register := ALU31; Y <0-31> = ALU <0-31>
RSQB	5	Load BREG with ALU shifted right one bit; Shift QREG right; Y <0-31> = ALU <0-31> LINK register := ALU 31
LSHB	6	Load BREG with ALU shifted left one bit; Y <0-31> = ALU <0-31> LINK gets ALU16, ALU0 for FLAG0 = 0,1 respectively.
LSQB	7	Load BREG with ALU shifted left one bit; Shift QREG left; Y <0-31> = ALU <0-31> LINK gets ALU16, ALU0 for FLAG0 = 0,1 respectively.

FPUD FIELD

Mnem	Value	Description
NLD	0	No load; Y < 0-63> = FPU < 0-63 >
QREG	1	Load QREG only; Y< 0-63> = FPU <0-63 >
BREG	2	Load BREG only: Y< 0-63> = FPU <0-63 >
AOUT	3	Load BREG only; Y< 0-63> = AREG <0-63 >
RSHB	4	Load BREG with FPU shifted right one bit; BREG < 0-63 > := 0, FPU < 0-62 > LINK := 0
RSQB	5	Load BREG, QREG with FPU, QREG shifted right one bit; BREG < 0-63 > :=0, FPU< 0-62 > QREG < 0-63 > :=FPU63, QREG <0-62> LINK := 0
LSHB	6	Load BREG with FPU shifted left one bit; BREG < 0-63 > := FPU <1-63 >, QREG0 LINK := FPU0
LSQB	7	Load BREG, QREG with FPU, QREG shifted left one bit; BREG < 0-63 > := FPU <1-63 > QREG0 QREG < 0-63 > := QREG <1-62> ,CRYY LINK := FPU0

In each case (for the ALUD field and the FPUD field) a zero value is a no load operation. Thus, when the FPU board is being utilized for a floating point operation by the system, the ALUD field has a zero value so that no ALU calculations can be loaded into any destination register, while the FPUD field assumes a value which identifies an appropriate destination register into which a floating point answer can be loaded in accordance with the FPUD field above. However, if data is being transferred between the ALU and the FPU, the microcode can be written to allow loading of a destination register on each board, according to the algorithm appropriate for the instruction being implemented.

In the same manner, when the ALU board is being utilized for a fixed point operation by the system, the FPUD field has a zero value so that no FPU calculations can be loaded into any destination register, while the ALUD field assumes a value which identifies an appropriate register into which an ALU result can be loaded in accordance with the ALUD field above.

Accordingly, substantially the same microcode is used for non-floating point operations. Floating point microcode is rewritten to control the FPU maintaining identical algorithms so that

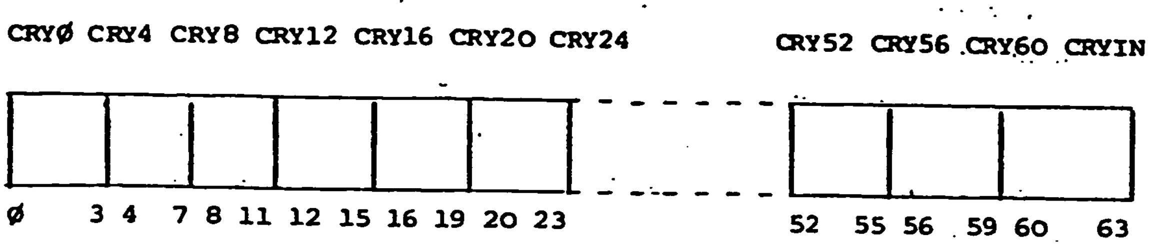
results with ALU only or ALU and FPU are the same. This requires an additional 2-bit field (the EXT field) in order to provide for suitable single or double precision floating point operation and an additional 3-bit field (the FPUD field) in order to identify FPU destination operations. Because the same microcode is used for each board, only a single microcontrol store is needed for storing the microinstructions required for controlling the operation of both boards. For double precision operation, the FPU board operates on a 64-bit word and, therefore, the overall double precision floating point calculation can be performed in less time than was required for the previous MV/8000 operation when the ALU board was used to provide both floating point single and a double precision calculations but could only do so by operating on 32-bit words at one time. However, the results produced are identical.

In addition to providing a separate board for performing single and double precision floating point computations, as discussed above, the floating point unit also is designed so as to provide "rounding" of the single or double precision floating point mantissa results in a unique manner as described below. As is well known, a conventional floating point mantissa calculation for addition or subtraction can occur in four cycles, as set forth below:

1. A comparison of the floating point operands is made to determine which is larger
2. The smaller operand is pre-scaled so as to align its radix point
3. The floating point operation is performed and normalized so as to produce an unrounded result
4. The unrounded result is rounded and the rounded result is normalized

In a floating point mantissa operation a round bit can be appropriately added into the least significant bit of the floating point result (e.g., bit 23 of the single precision operation or bit 55 of a double precision operation). Techniques for performing such rounding operation are discussed in our copending European Patent Application No. 82302080.5. As discussed therein the calculation is performed in 4-bit slices with appropriate "carry" bits for each slice being effectively inserted into the next higher adjacent slice, as shown below, (CRYØ is the carry bit for the overall result). In

an actual circuit implementation, for example, carry look-ahead circuits can be used to supply the carry-in bits to each slice.



The floating point computation can be performed by using a bit-slice integrated circuit which is available under the designation 2901 made by several manufacturers (e.g., IDM 2901A-1 IC ship made by National Semiconductor Corporation of Santa Clara, California). An "AND" operation (for a 4-bit slice) performed by such well-known circuit can be described as follows. Let the bits of the 4-bit slice operands be designated as R_0, R_1, R_2, R_3 and S_0, S_1, S_2 and S_3, respectively. For such AND operation, the propagate ($\overline{P}$)output is "low" and the generate ($\overline{G}$) output is defined by the Boolean equation:

$$\overline{G} = \overline{R_0S_0+R_1S_1+R_2S_2+R_3S_3}$$

and the carry out (CRYOUT) output is defined by the Boolean equation:

$$CRYOUT = CRYIN + R_0S_0+R_1S_1+R_2S_2+R_3S_3$$

The rounding operation to be described in accordance with the invention herein makes use of the above-described operation of the 2901 IC circuit in providing a rounding technique which avoids the extra time normally required in conventional rounding techniques which effectively require a multiplexing operation when inserting the round bit at the proper slice (as discussed in the aforesaid co-pending application).

Thus, the invention recognizes that in the above operation of the 2901 unit, if one of the operands is zero and the unit is performing an AND operation the carry in bit is propagated through the slice as the carry out bit so that

$$CRYOUT = CRYIN$$
$$P = 1 \ (\overline{P} \text{ is low})$$
$$G = Ø \ (\overline{G} \text{ is high})$$

That is, whatever is inserted as a CRYIN bit to the 4-bit slice effectively from the next lower adjacent 4-bit slice is propagated directly through the slice and is provided as the CRYOUT

bit thereof. In an actual circuit implementation, for example, this propagation can be accomplished by carry look-ahead circuits. Carry look-ahead circuits will propagate a carry (P=1) bit, but not generate one (G=Ø).

Therefore, in order to add the round bit into the appropriate bit position, bit 23 for a single precision mantissa computation and bit 55 for a double precision mantissa computation, the round bit (suitably calculated as described, for example, in the aforementioned copending application) can be inserted at bit 63 (the least significant bit) as the CRYIN bit of the lower 4-bit slice.

For a single precision mode for each of the middle eight 4-bit slices (bits 24-55) and the lower two 4-bit slices (bits 56-63), one of the operands is set to zero and each is controlled to perform an AND operation so that whatever is inserted as a CRYIN bit at bit 63 is propagated effectively through each of such slices as a CRYOUT bit.

Accordingly, if the round bit is added as the "carry in" bit CRYIN, such round bit is propagated effectively through the lower two and the middle eight 4-bit slices so as to be added as the carry in bit CRY24 at bit 23 of 4-bit slice 20-23 (the lowest 4-bit slice of the unrounded result) so that the rounding operation is performed without the need to utilize an effective multiplexing operation at bit 23.

In the same manner, for a double precision floating point mantissa computation, the lower two slices (bits 56-63) are arranged during the rounding cycle to have one of their operands set to zero and are controlled to perform AND operations. A round bit inserted as the CRYIN bit at bit 63 is then effectively propagated through the lower two slices and inserted as the carry in bit CRY56 (the round bit) at bit 55 of the lower 4-bit slice (bits 52-55) of the unrounded result to complete the rounding of the final 56-bit floating point mantissa result.

In the same manner, during non-rounding cycles in a single precision operation, the CRYIN bit set by microcode is effectively propagated to bit 23 as described above, since according to TABLE 1 the lower ten slices have one of their operands set to zero and are controlled to perform AND operations.

Specific logic which is used to implement the floating point

unit board of Fig. 3 and 4 is shown in Figs. 5-36. Thus, the 4-bit microprocessor slice logic is depicted in Figs. 5-12. Figs. 13-16 show the 4-bit slice control logic and the control store mode (CSM) and extension (EXT) decode logic. Figs. 17-18 show various logic such as the look-ahead carry logic, floating point status register logic, test logic, random field logic and the central processor data (CPD) bus control logic. Figs. 19-20 depict logic associated with the operation of the multiply register (MREG) and multiply accumulator (MACC) of Figs. 3 and 4. Figs. 21-23 show logic associated with the T-latch unit and the CPD bus interface logic depicted in Figs. 3 and 4. Figs. 24-26 show logic associated with the coarse nibble rotator, the Z-latch logic and the shift enable logic of Figs. 3 and 4. Figs. 27-29 show logic associated with the fine nibble rotator of Figs. 3 and 4 and further shift control logic. The nibble shift multiplexer units and the processor memory data bus interface logic are shown in Figs. 30-32. D-bus enable logic, dispatch logic, normalization logic and the round logic for generating the round bit $\overline{\text{SET ROUND}}$ are depicted in Figs. 33-34. The ACSR (source register), ACDR (destination register), A-register and B-register logic of Figs. 3 and 4 are shown in Figs. 35-36, as well as various clock logic required for FPU operation. The use by those in the art of such specific logic as depicted in Figs. 5-36, together with the foregoing description, will permit one in the art to practise the invention and to fabricate an FPU board thereof in accordance with the invention, as set forth above.

TABLE I

CSM & EXT - Control Store Mode & Extension Fields

CSM		EXT		Half-cycle 1			Half-cycle 2		
Mnem	Val	Mnem	Val	ALUS	ALUOP	FPUD	ALUS	ALUOP	FPUD
SMATH	0								
SFIXP	1	DBL	0	uI16	uI16	#	DZ16	OR16	uI
SGEN	2	SGL	2	uI6/DZ10	uI6/AND10	#	DZ16	ORG/AND10	uI
SATU	3								
FMATH	4								
FFIXP	5	DBL	0	uI16	uI16	#	uI16	uI16	uI
FGEN	6	SGL	2	uI6/DZ10	uI6/AND10	#	uI6/DZ10	uI6/AND10	uI
FATU	7								
BOUT	A								
QDEC	C	DBL	0	uI16	uI16	#	uI16	uI16	uI
QINC	D	SGL	2	uI6/DZ10	uI6/AND10	#	uI6/DZ10	uI6/AND10	uI
QADD	E								
MPY	8	NOP	0	@16	@16	#	@16	@16	uI
MPY	8	FST	1	@16	@16	#	@16	@16	uI
DIV	9	NOP	0	uI16	@16	uI	uI16	@16	uI
NORM	B	DBL	0	uI16	uI16	#	DZ16	OR16	uI
NORM	B	ADS	1	uI6/ZQ10	uI16	#	DZ16	OR16	uI
NORM	B	SGL	2	uI6/DZ10	uI6/AND10	#	DZ16	OR16	uI
PRESC	F	WTR	0	@16	uI16	#	DZ16	OR16	uI
PRESC	F	NTR	1	@16	uI16	#	DZ16	OR16	uI

In the tables the following abbreviations are used:

uI - Represents the u-order from the appropriate field of the specified instruction.

uI16 - The u-order from the appropriate field of the specified u-instruction controls all 16 slices.

<specified>16 - The specified u-order from the above table controls all 16 slices.

- No clock takes place, but the FPUD u-order still controls the AOUT multiplexer, i.e. <0-31> still gets the value centraled by uI.

@ (@16) - The u-order will defer to a predecoded or "Forced" value (for all 16 slices).

uI6/<specified>10 - The u-order from the appropriate field of the specified u-instruction controls the most significant 6 slices. The least significant 10 slices are forced to the <specified> u-order from the above table.

TABLE 2

Mnem	Val	Half-cycle 1 ALUS	ALUOP	ALUD	Half-cycle 2 ALUS	ALUOP	ALUD	DIST Type	CRYIN	RAND Type
SMATH	0	uI	uI	#	DZ	OR	uI	Math	Type0	Math
SFIXP	1	uI	uI	#	DZ	OR	uI	Gen	Type1	Fixp
SGEN	2	uI	uI	#	DZ	OR	uI	Gen	Type0	Gen
SATU	3	uI	uI	#	DZ	OR	uI	Gen	Type0	Atu
FMATH	4	uI	uI	#	uI	uI	uI	Math	Type0	Math
FFIXP	5	uI	uI	#	uI	uI	uI	Gen	Type1	Fixp
FGEN	6	uI	uI	#	uI	uI	uI	Gen	Type0	Gen
FATU	7	uI	uI	#	uI	uI	uI	Gen	Type0	Atu
MPY	8	@	@	#	@	@	uI	Math	Type2	Math
DIV	9	uI	@	uI	uI	@	uI	Math	Type3	Math
BOUT	A	uI	uI	#	ZB	OR	uI	Gen	Type0	Gen
NORM	B	uI	uI	#	DZ	OR	uI	Math	Type0	Math
QDEC	C	ZQ	SUB	QREG	uI	uI	uI	Gen	*Type0	Gen
QINC	D	ZQ	ADD	QREG	uI	uI	uI	Gen	*Type0	Gen
QADD	E	DQ	ADD	QREG	uI	uI	uI	Gen	*Type0	Gen
PRESC	F	@	OR	#	DZ	OR	uI	Math	Type0	Math

* - The CRYIN is forced to a zero in the first half cycle in modes QDEC and QADD, and to a one during the first half of mode Qinc.

CLAIMS

1. A data processing system handling both fixed point and floating point computation operations, comprising an arithmetic unit for profiding fixed point arithmetic computations, a floating point unit for providing floating point arithmetic computations and a single control store means for supplying microinstructions to control the operations of both the units, characterised in that both of the units are responsive to a plurality of commonly shared control fields of the microinstructions during their respective operations and in that the floating point unit is further responsive to at least one additional control field during its operation.

2. A system in accordance with claim 1, characterised in that the arithmetic computation unit is responsive to a selected value of a first control field for preventing the use of any fixed point computation results from the arithmetic unit when the floating point unit is supplying its floating point computation result to the data processing system.

3. A system in accordance with claim 1 or 2, characterised in that the floating point unit is responsive to a selected value of a second control field for preventing the use of any floating point computation results from the floating point unit when the arithmetic unit is supplying its fixed point computation result to the data processing system.

4. A system in accordance with claim 3, the first and second control fields are fields which identify respectively the destination of a fixed point computation result from the arithmetic unit and the destination of a floating point computation result from the floating point unit.

5. A system in accordance with claim 4, characterised in that the selected value of the arithmetic destination control field is set for a no-load operation when the floating point unit is supplying its result to the data processing system and the selected value of the floating point destination control field is set for a no-load operation when the arithmetic unit is supplying its result to the data processing system.

6. A system in accordance with any of claims 1 to 5, characterised in that a common control field of the control store is used to identify the source of data which is to be supplied either to the arithmetic unit when it is computing a fixed point result or to the floating point unit when it is computing a floating point result.

7. A system in accordance with claim 6, characterised in that a common control field of the control store is used to identify the operation which is to be performed by either the arithmetic unit when it is performing a fixed point operation or the floating point unit when it is performing a floating point operation.

8. A system according to any of claims 1 to 7, characterised in that the said at least one additional control field comprises a control field for identifying whether the floating point unit is operating on data words of a first or a second number of data bits.

9. A system in accordance with claim 8, characterised in that the first number is 32 bits for providing a 24 bit floating point mantissa result and the second number is 64 bits for providing a 56 bit floating point mantissa result.

10. A system in accordance with claim 9, characterised in that, when the control field identifies a 32 bit operation, the field controls the operation of the floating point unit in a manner such that the operands of the computation used for the first 24 bits of the floating point words are controlled by a microinstruction from the control store and the remaining 40 bits of the floating point words are zero.

11. A data processing system having a floating point unit for providing floating point computation operations, the floating point unit including processing means for providing an overall computation result comprising a first plurality of bits representing a desired floating point computation result and a second plurality of bits not used in this desired computation result; and means for providing a carry-in bit to be added to the floating point computation result;

characterised by means for adding the carry-in bit to the least significant bit of the overall computation result and for causing the carry-in bit to be effectively propagated through the processing means so as to be added to the least significant bit of the first plurality of bits to form the floating point computation result.

12. A system in accordance with claim 11, characterised in that the processor means comprises a plurality of multiple-bit slice processor units each processing selected bit slices of floating point operands to provide the overall computation result, the overall result being formed of a plurality of bit slices each effectively provided with a carry-in bit from an adjacent lower significant bit slice processor unit and effectively supplying a carry-out bit to an adjacent higher significant bit slice processor unit, the carry-in bit added to the least significant bit of the overall result being effectively propagated through the plurality of bit slice processors which form the second plurality of bits so as to be added to the bit slice which forms the least significant bit slice of the first plurality of bits.

13. A system in accordance with claim 12, characterised in that the processing means has an operating characteristic such that when a bit slice processor unit thereof is placed in an AND operating condition and at least one of the operands of the bit slice processor unit is set to zero, the carry-in bit is effectively propagated through the bit slice processor unit as the carry-out bit thereof.

14. A system in accordance with claim 13, characterised in that each bit slice processor unit is capable of asserting a propagate bit or a generate bit and the means for adding the carry-in bit and for causing the carry-in bit to be effectively propagated through the bit slice processor unit includes a plurality of look-ahead logic means responsive to the bit slice processor units whereby, when a bit slice processor unit asserts its propagate bit but does not assert its generate bit, the look-ahead logic means associated therewith propagates its carry-in bit but does not generate a carry-out bit.

15. A system in accordance with claim 13 or 14, characterised in that each of the bit slice processor units which form the second plurality of bits is placed in an AND operating condition and at least one of the floating point operands in the bit slice processor units is set to zero, whereby the carry-in bit is effectively propagated through each said bit slice thereof, the carry-in bit being added to the least significant bit of the first plurality of bits to form the floating point computation result.

16. A system in accordance with any of claims 12 to 15, characterised in that the bit slice processor units each process 4 bits of the floating point operands.

17. A system in accordance with any of claims 12 to 16, characterised in that the overall floating point computation result comprises 64 bits.

18. A system in accordance with claim 17, characterised in that the first plurality of bits comprises 24 bits and the second plurality of bits comprises 40 bits.

19. A system in accordance with claim 17, characterised in that the first plurality of bits comprises 56 bits and the second plurality of bits comprises 8 bits.

20. A system in accordance with any of claims 11 to 19, characterised in that the floating point computation result is an unrounded result and the means providing the carry-in bit comprises means for providing a round bit as the carry-in bit which is added to the least significant bit of the overall computation result, the round bit thereby being added to the unrounded result to produce a rounded floating point computation result.

21. A system in accordance with any of claims 11 to 19, characterised in that the means providing the carry-in bit comprises means responsive to a carry-in bit supplied from a source external to the floating point computation unit as the carry-in bit which is added to the least significant bit of the overall computation result,

the externally supplied carry-in bit thereby being added to the floating point computation result.

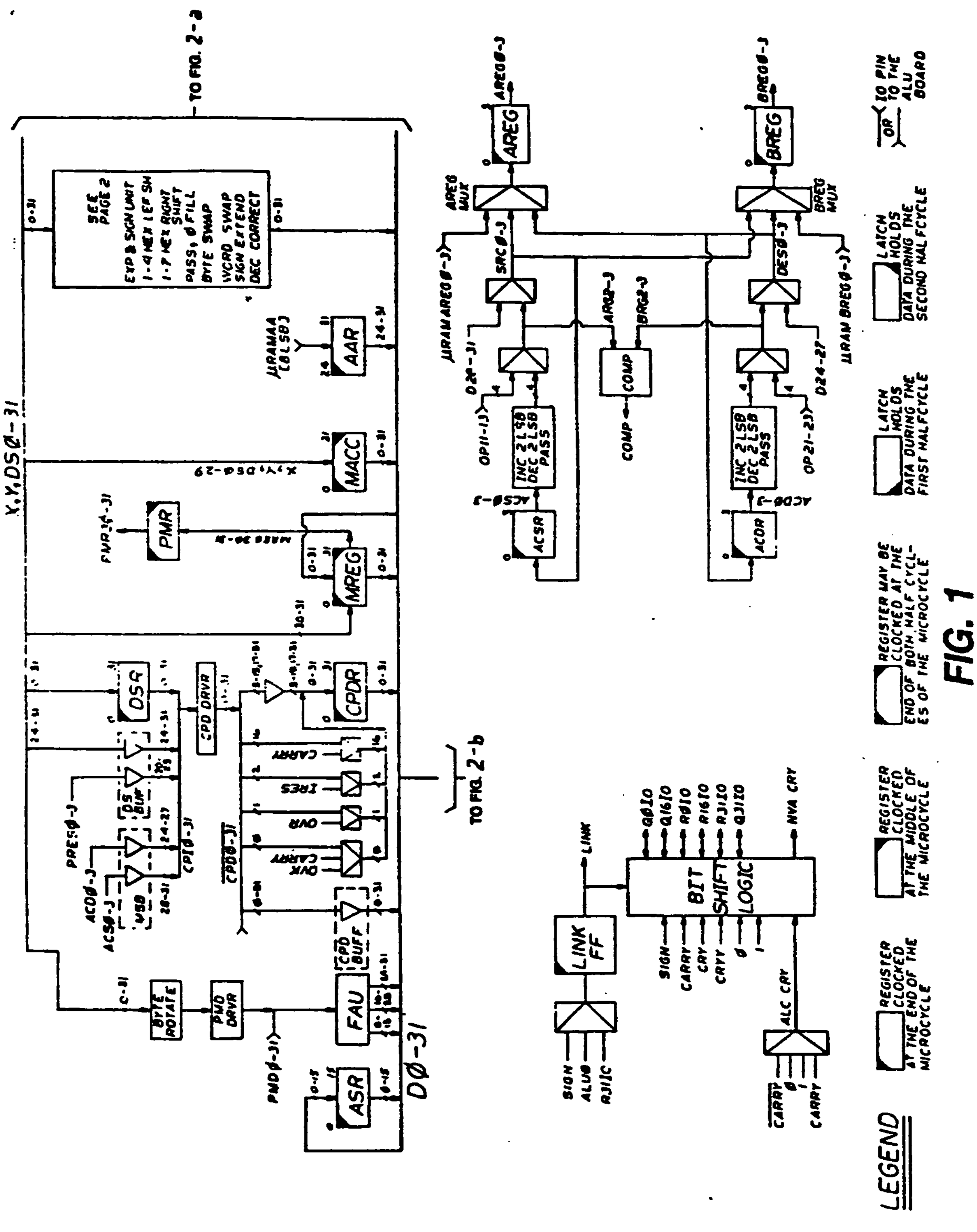

TO FIG. 2-a
X,Y,DSØ-31
SEE PAGE 2
EXP & SIGN UNIT
1-4 HEX LEF SH
1-7 HEX RIGHT SHIFT
PASS, Ø FILL
BYTE SWAP
WORD SWAP
SIGN EXTEND
DEC CORRECT
AAR
MACC
PMR
MREG
DSR
CPD DRVR
CPDR
CARRY
IRES
OVR
OVR CARRY
CPDØ-31
DS BUF
PRESØ-3
ACDØ-3
ACSØ-3
BYTE ROTATE
PWD DRVR
CPD BUFF
FAU
ASR
DØ-31
TO FIG. 2-b
AREG
AREG MUX
BREG
BREG MUX
SRCØ-3
DESØ-3
COMP
ACSR
ACDR
INC 2 LSB
DEC 2 LSB
PASS
LINK
LINK FF
BIT SHIFT LOGIC
SIGN
CARRY
ALC CRY
NYA CRY
LEGEND
REGISTER CLOCKED AT THE END OF THE MICROCYCLE
REGISTER CLOCKED AT THE MIDDLE OF THE MICROCYCLE
REGISTER MAY BE CLOCKED AT THE END OF BOTH HALF CYCLES OF THE MICROCYCLE
LATCH HOLDS DATA DURING THE FIRST HALF CYCLE
LATCH HOLDS DATA DURING THE SECOND HALF CYCLE
TO PIN TO THE ALU BOARD

FIG. 1

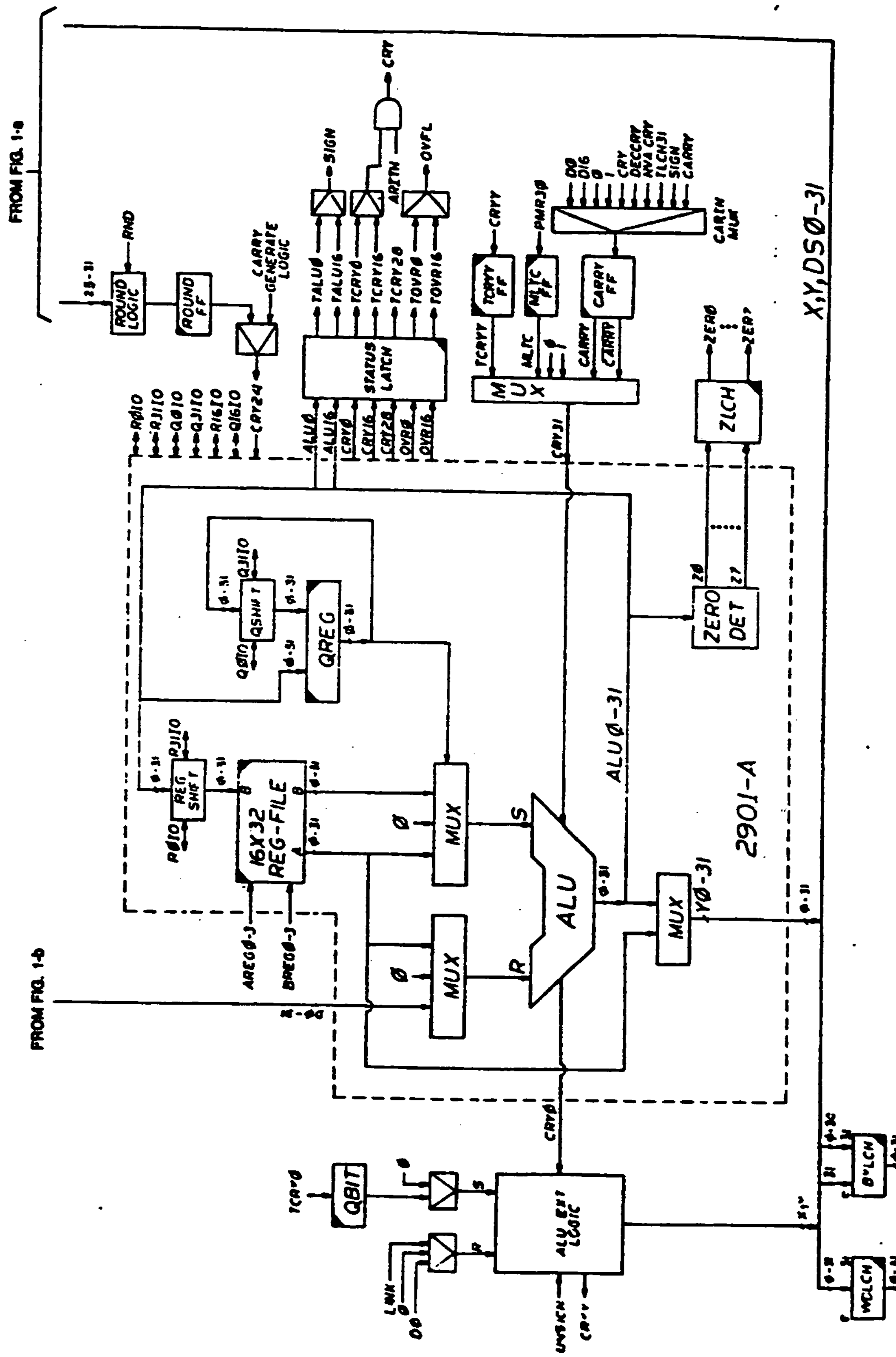

FROM FIG. 1-B
FROM FIG. 1-B
ROUND LOGIC
RND
ROUND FF
CARRY GENERATE LOGIC
CRY24
STATUS LATCH
TALUØ
TALU16
TCRYØ
TCRY16
TCRY20
TOVRØ
TOVR16
SIGN
ARITH
OVFL
CRY
TCRYY FF
MLTC FF
CARRY FF
CARIN MUX
TCRYY
PMRJØ
ZLCH
ZERO DET
QREG
QSHFT
16X32 REG-FILE
REG SHFT
MUX
ALU
ALUØ-31
2901-A
YØ-31
X,Y,DSØ-31
AREGØ-3
BREGØ-3
CRYØ
QBIT
ALU EXT LOGIC
WCLCH
LAØ-31

FIG. 2

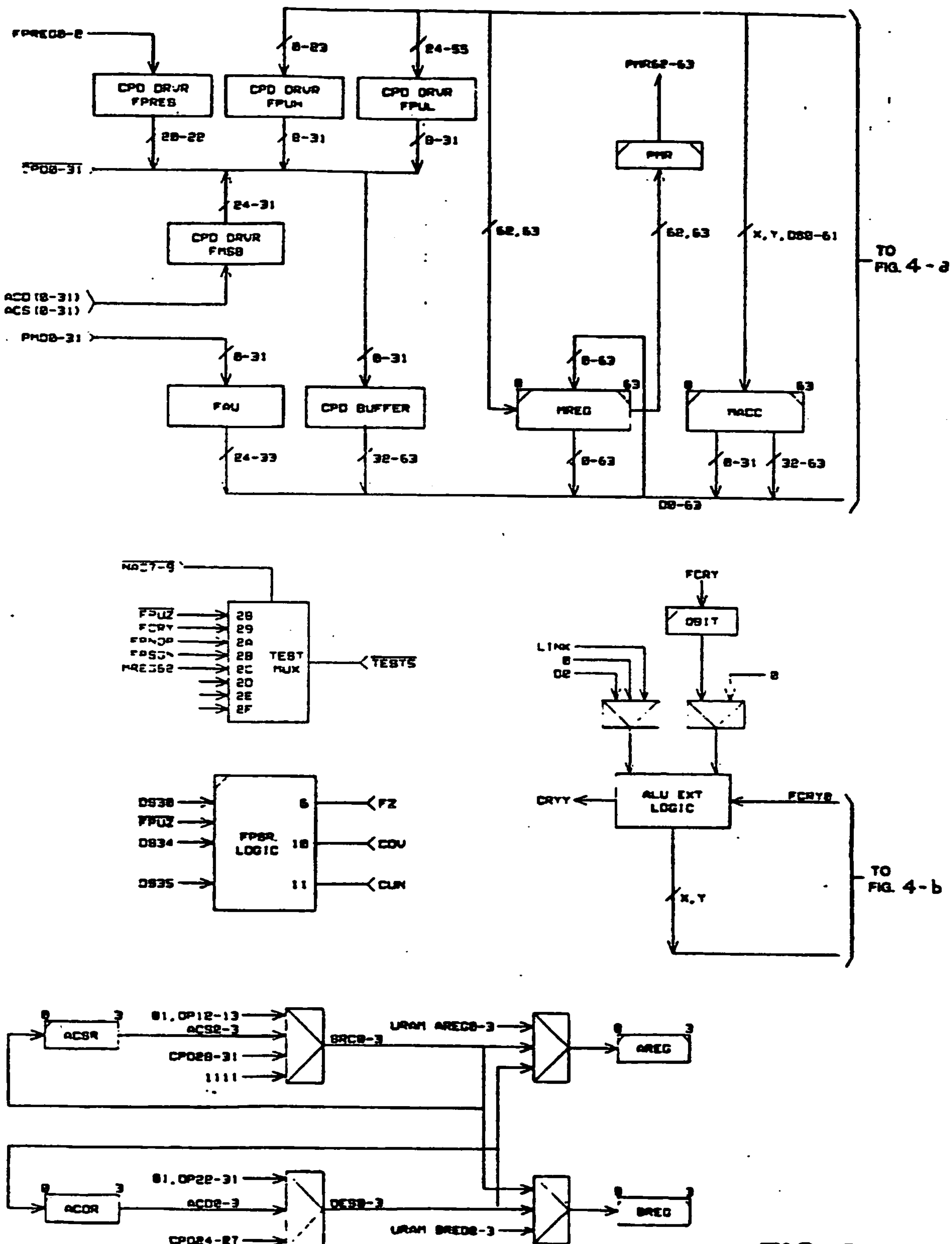

FPREG0-2
CPD DRVR FPREG
CPD DRVR FPUM
CPD DRVR FPUL
0-23
24-55
20-22
0-31
0-31
CPD0-31
PMR62-63
PMR
24-31
CPD DRVR FMSB
62,63
62,63
X,Y,DS0-61
TO FIG. 4-a
ACD(0-31)
ACS(0-31)
PMD0-31
0-31
0-31
0-63
FAU
CPD BUFFER
MREG
MACC
24-33
32-63
0-63
0-31
32-63
D0-63
FCRY
DBIT
LINK
CRYY
ALU EXT LOGIC
FCRY0
X,Y
TO FIG. 4-b
TEST MUX
TESTS
FPUZ
FCRY
FPSR LOGIC
DS38
DS34
DS35
FZ
COV
CUN
ACSR
ACDR
AREG
BREG
URAM AREG0-3
URAM BREG0-3
CPD28-31
CPD24-27
1111
ACD0-3
ACS0-3

FIG. 3

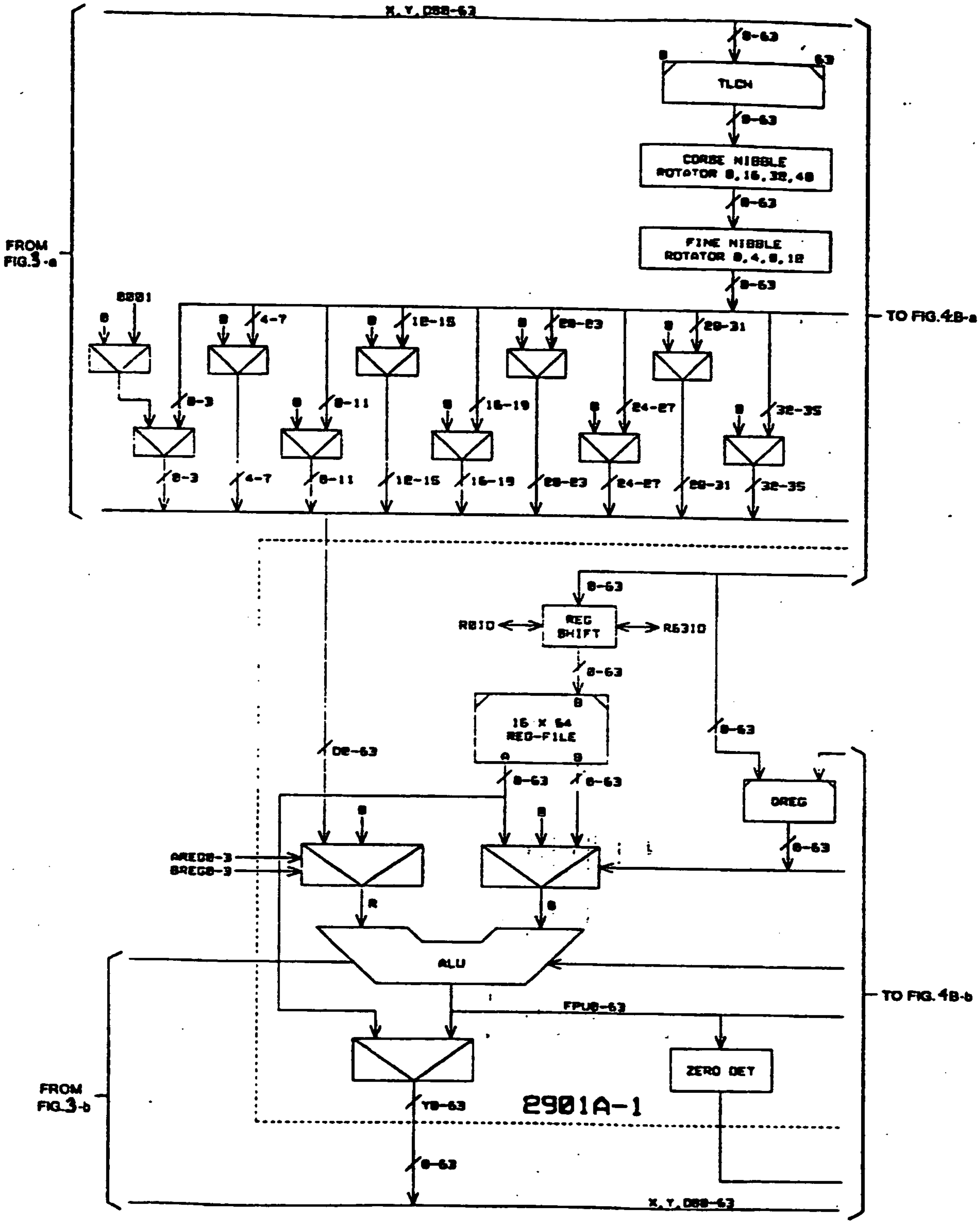
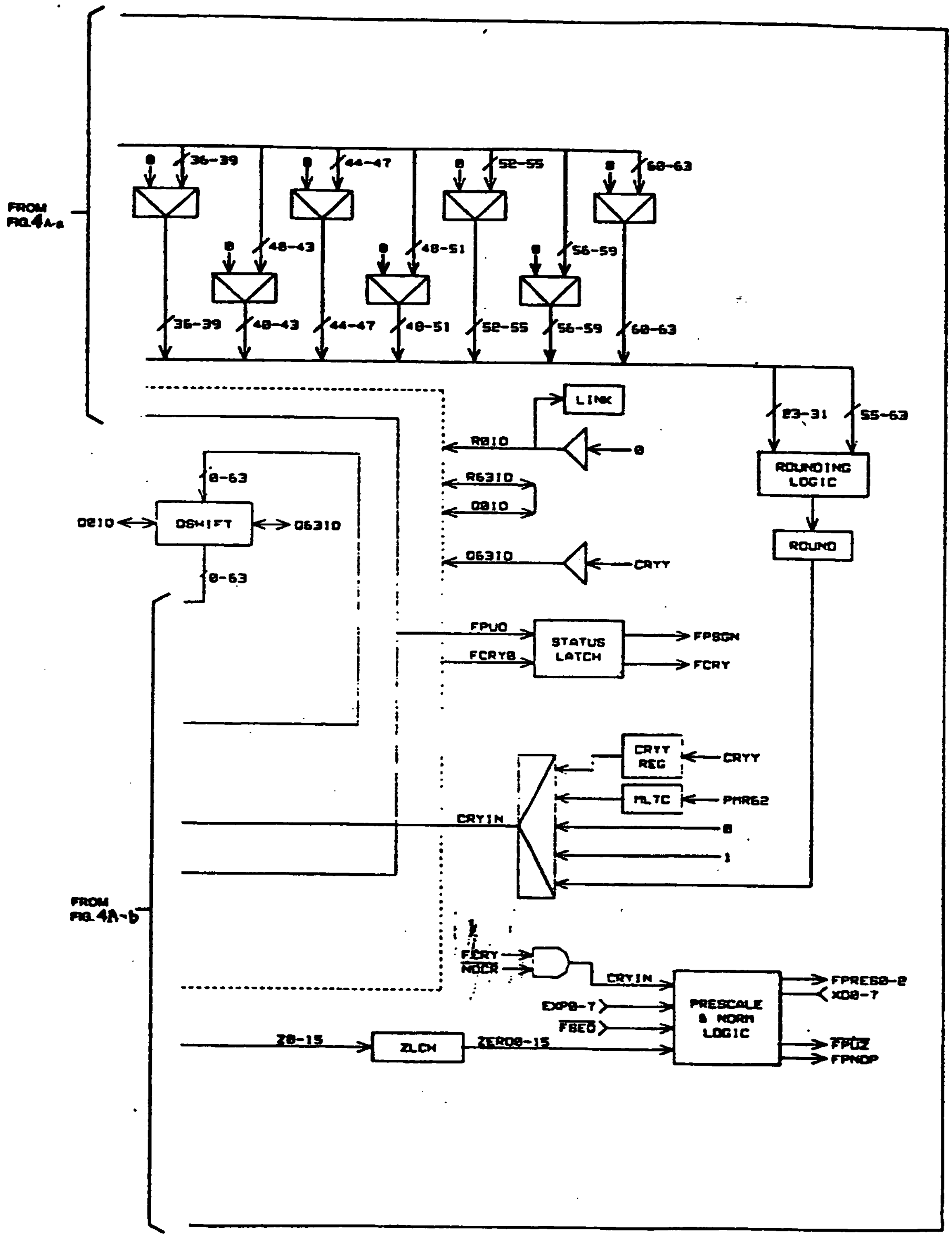

X, Y, DBB-63
TLCH
CORSE NIBBLE ROTATOR 0,16,32,48
FINE NIBBLE ROTATOR 0,4,8,12
FROM FIG.3-a
TO FIG.4B-a
REG SHIFT
R0ID
R631D
16 X 64 REG-FILE
DREG
AREG0-3
BREG0-3
ALU
FPU0-63
ZERO DET
2901A-1
FROM FIG.3-b
TO FIG.4B-b

FIG. 4A



FROM FIG.4A-a
0
36-39
44-47
52-55
60-63
48-43
48-51
56-59
LINK
R8IO
R63IO
D8IO
D63IO
CRYY
0-63
D8IO
DSHIFT
D63IO
23-31
55-63
ROUNDING LOGIC
ROUND
FPUO
FCRY8
STATUS LATCH
FPSGN
FCRY
CRYY REG
MLTC
PMR62
CRYIN
1
FROM FIG.4A-b
FCRY
NOCR
PRESCALE & NORM LOGIC
FPRES0-2
XD0-7
EXP0-7
FSEO
FPUZ
FPNOP
Z0-15
ZLCH
ZERO0-15

FIG.4B

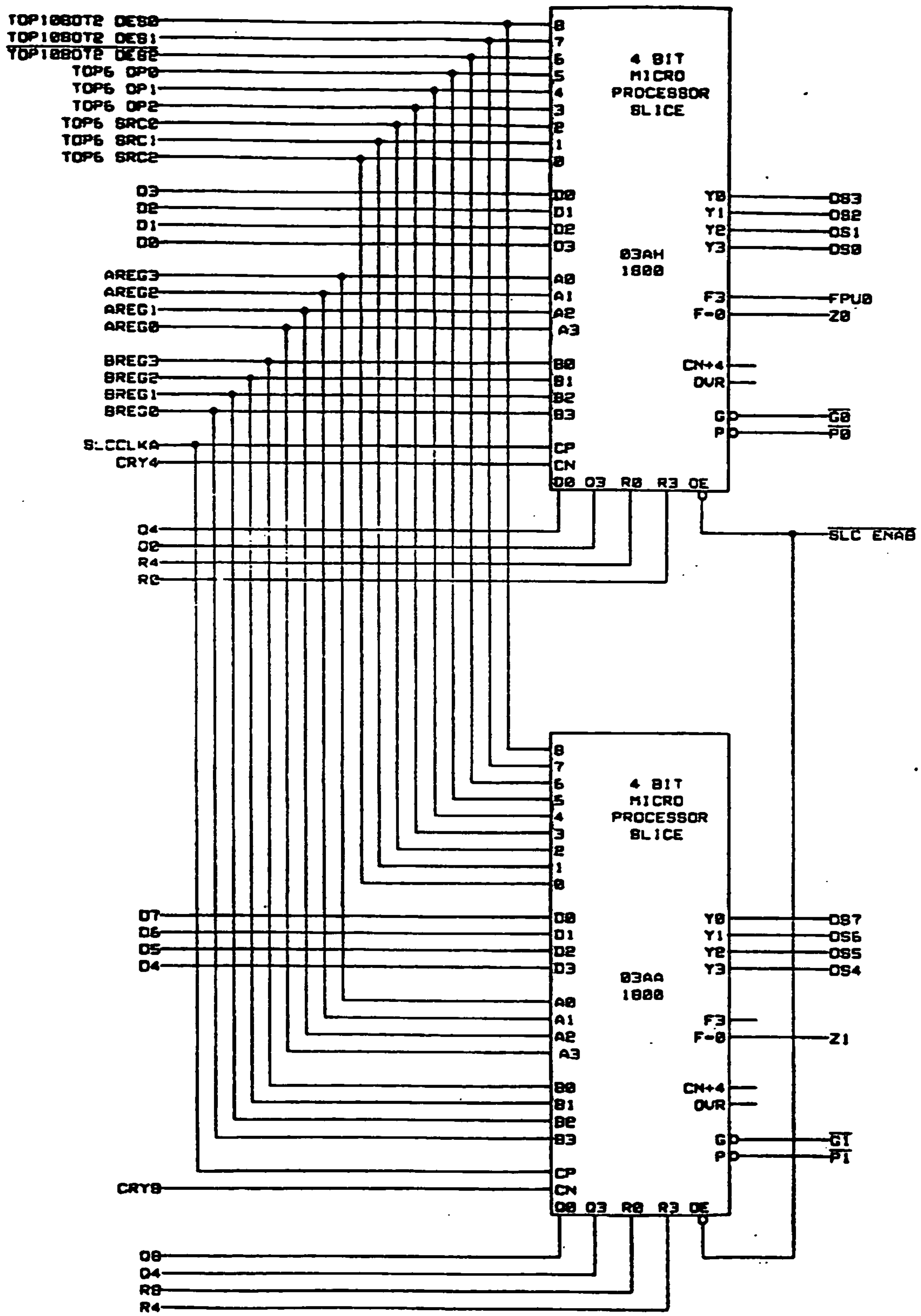

4 BIT
MICRO
PROCESSOR
SLICE
83AH
1800
4 BIT
MICRO
PROCESSOR
SLICE
83AA
1800
SLC ENAB
CRY4
CRY8
SLCCLKA

FIG. 5

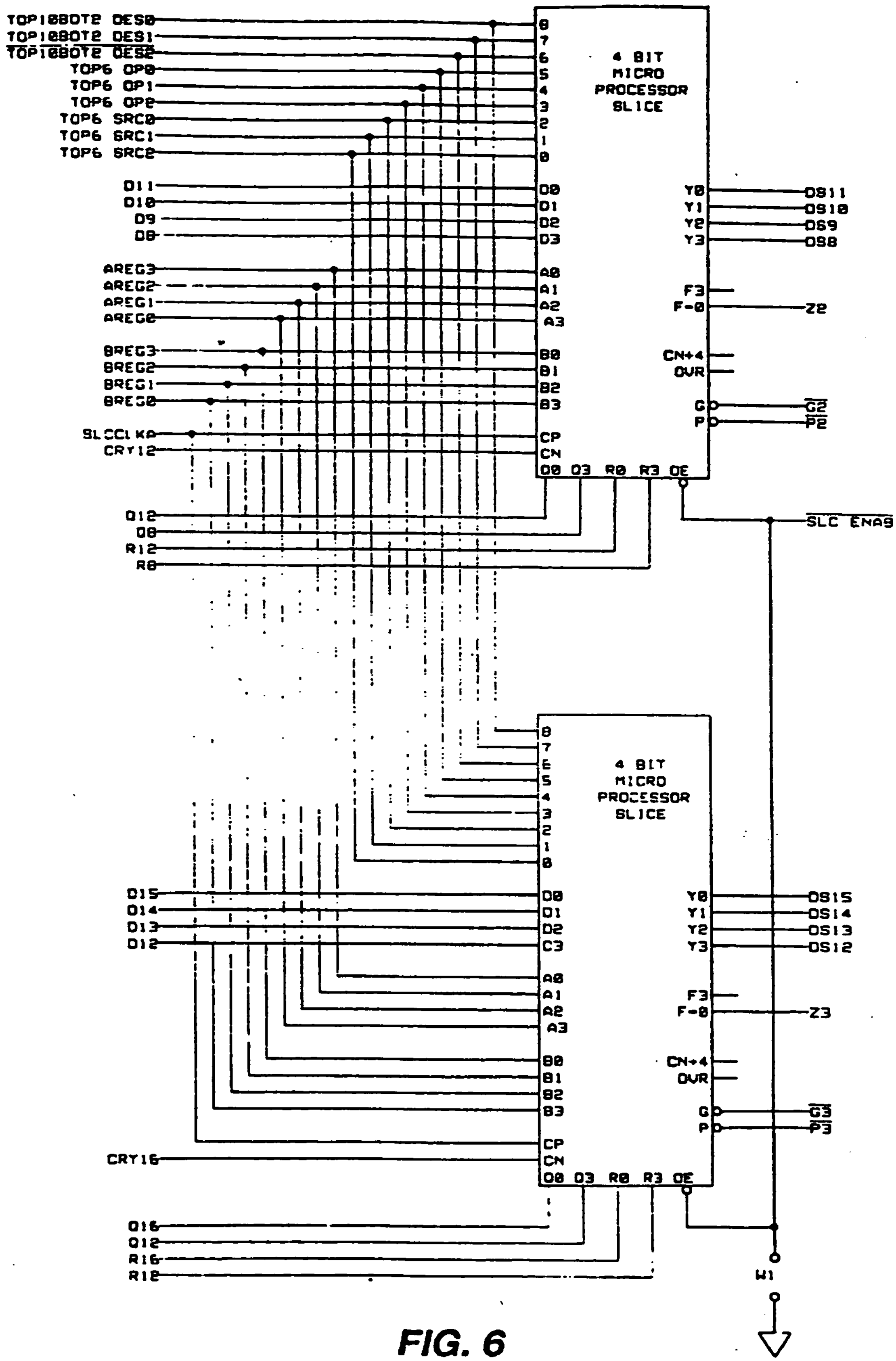

TOP10BOT2 DES0
TOP10BOT2 DES1
TOP10BOT2 DES2
TOP6 OP0
TOP6 OP1
TOP6 OP2
TOP6 SRC0
TOP6 SRC1
TOP6 SRC2
D11
D10
D9
D8
AREG3
AREG2
AREG1
AREG0
BREG3
BREG2
BREG1
BREG0
SLCCLKA
CRY12
4 BIT MICRO PROCESSOR SLICE
DS11
DS10
DS9
DS8
Z2
G2
P2
D12
D8
R12
R8
SLC ENAS
D15
D14
D13
D12
CRY16
4 BIT MICRO PROCESSOR SLICE
DS15
DS14
DS13
DS12
Z3
G3
P3
D16
D12
R16
R12
W1

FIG. 6

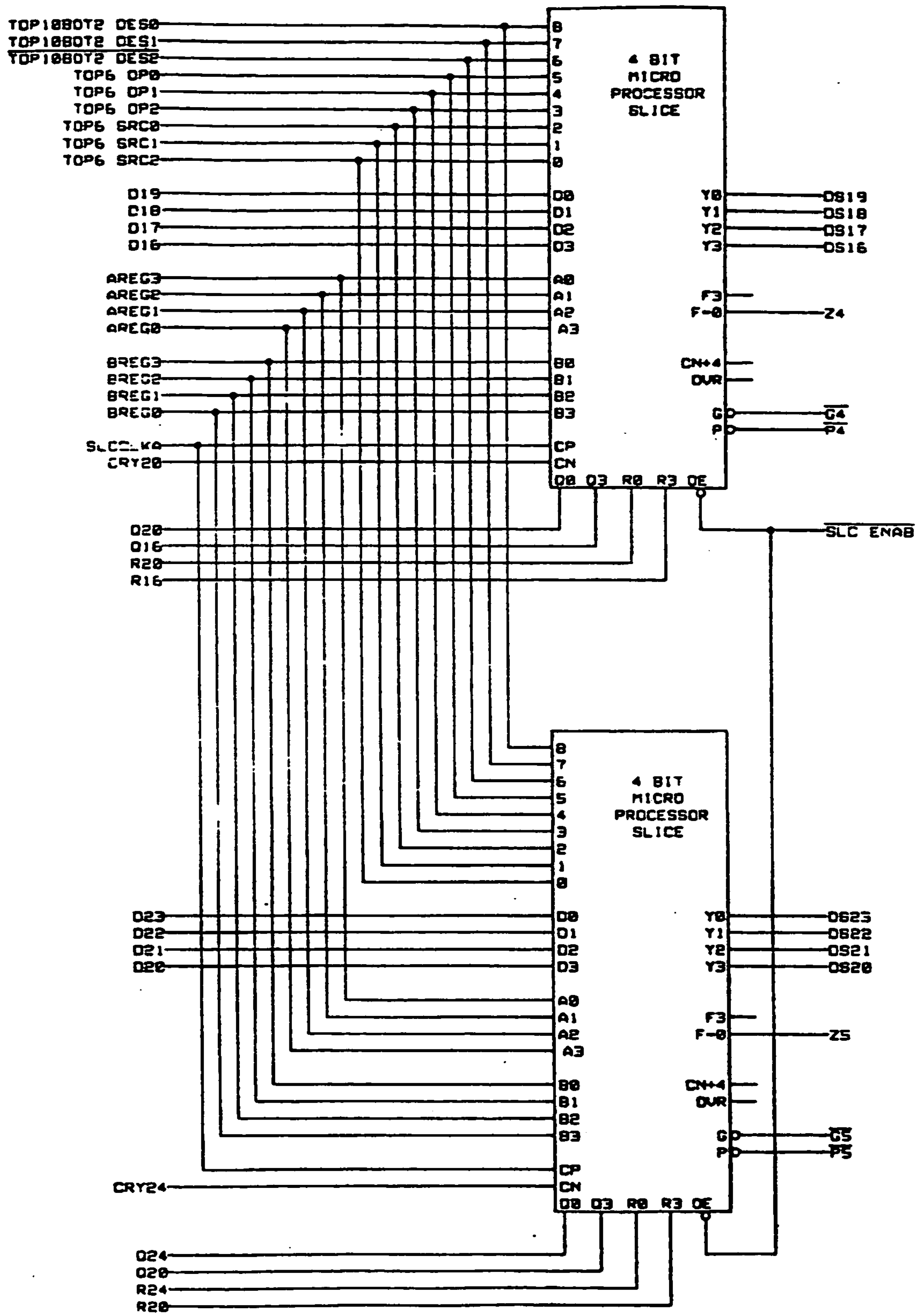
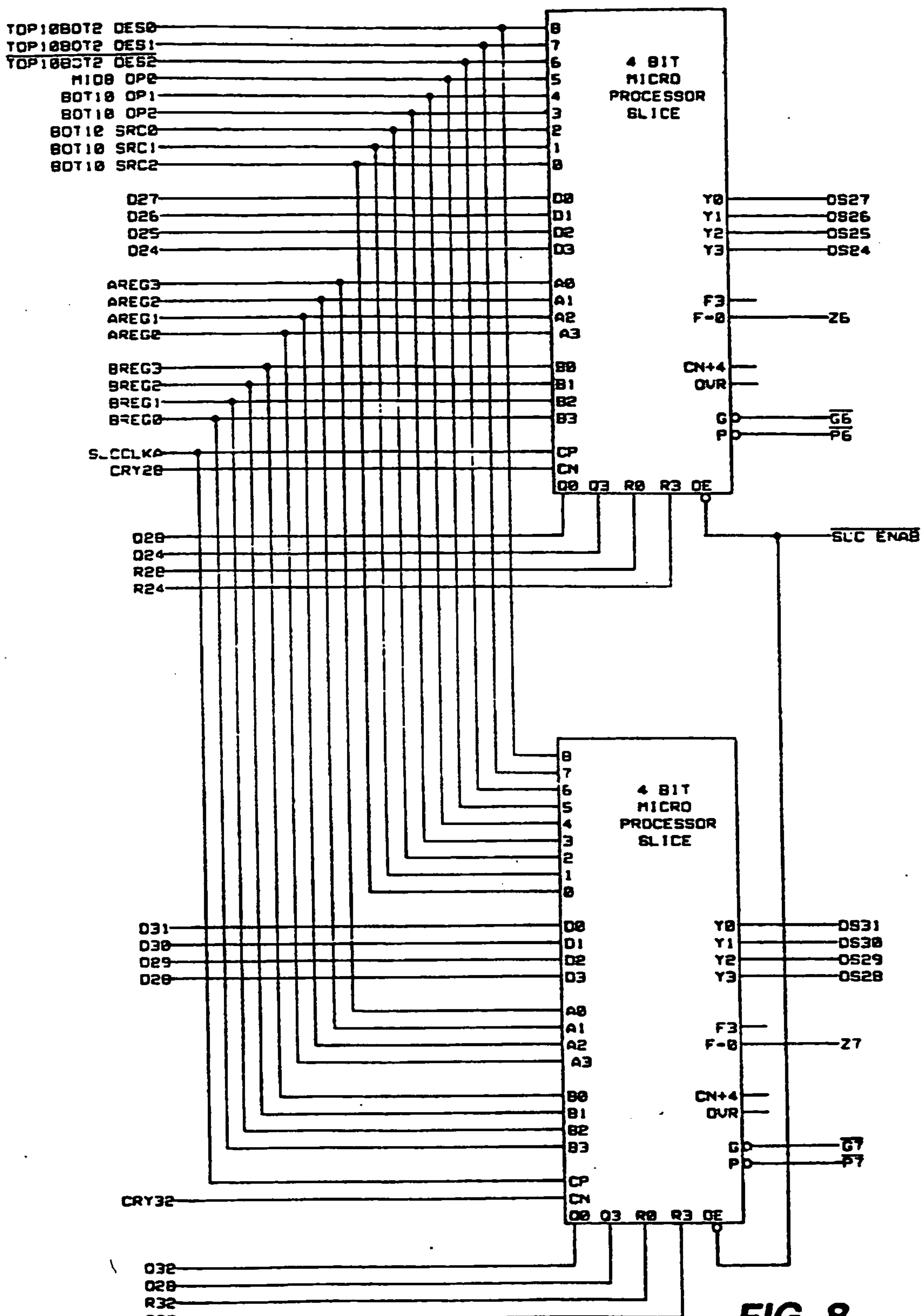

TOP10BOT2 DES0
TOP10BOT2 DES1
TOP10BOT2 DES2
TOP6 OP0
TOP6 OP1
TOP6 OP2
TOP6 SRC0
TOP6 SRC1
TOP6 SRC2
D19
D18
D17
D16
AREG3
AREG2
AREG1
AREG0
BREG3
BREG2
BREG1
BREG0
SLCCLKA
CRY20
4 BIT MICRO PROCESSOR SLICE
DS19
DS18
DS17
DS16
Z4
G4
P4
SLC ENAB
D20
D16
R20
R16
4 BIT MICRO PROCESSOR SLICE
D23
D22
D21
D20
DS23
DS22
DS21
DS20
Z5
G5
P5
CRY24
D24
D20
R24
R20

FIG. 7



TOP10BOT2 DES0
TOP10BOT2 DES1
TOP10BOT2 DES2
MID8 OP0
BOT10 OP1
BOT10 OP2
BOT10 SRC0
BOT10 SRC1
BOT10 SRC2
D27
D26
D25
D24
AREG3
AREG2
AREG1
AREG0
BREG3
BREG2
BREG1
BREG0
SLCCLKA
CRY28
Q28
Q24
R28
R24
4 BIT MICRO PROCESSOR SLICE
OS27
OS26
OS25
OS24
Z6
G6
P6
SLC ENAB
D31
D30
D29
D28
CRY32
Q32
Q28
R32
R28
4 BIT MICRO PROCESSOR SLICE
OS31
OS30
OS29
OS28
Z7
G7
P7

FIG. 8

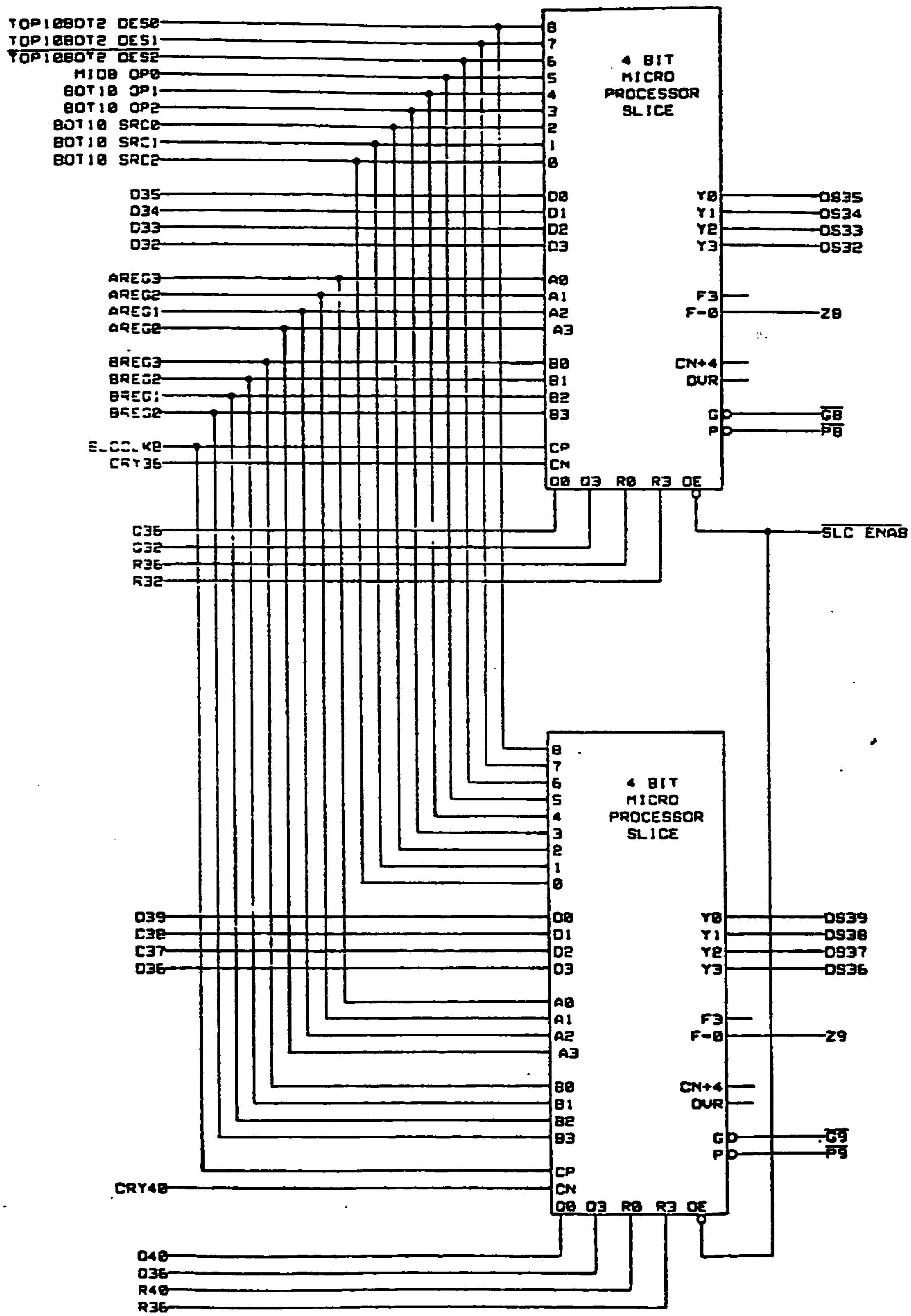
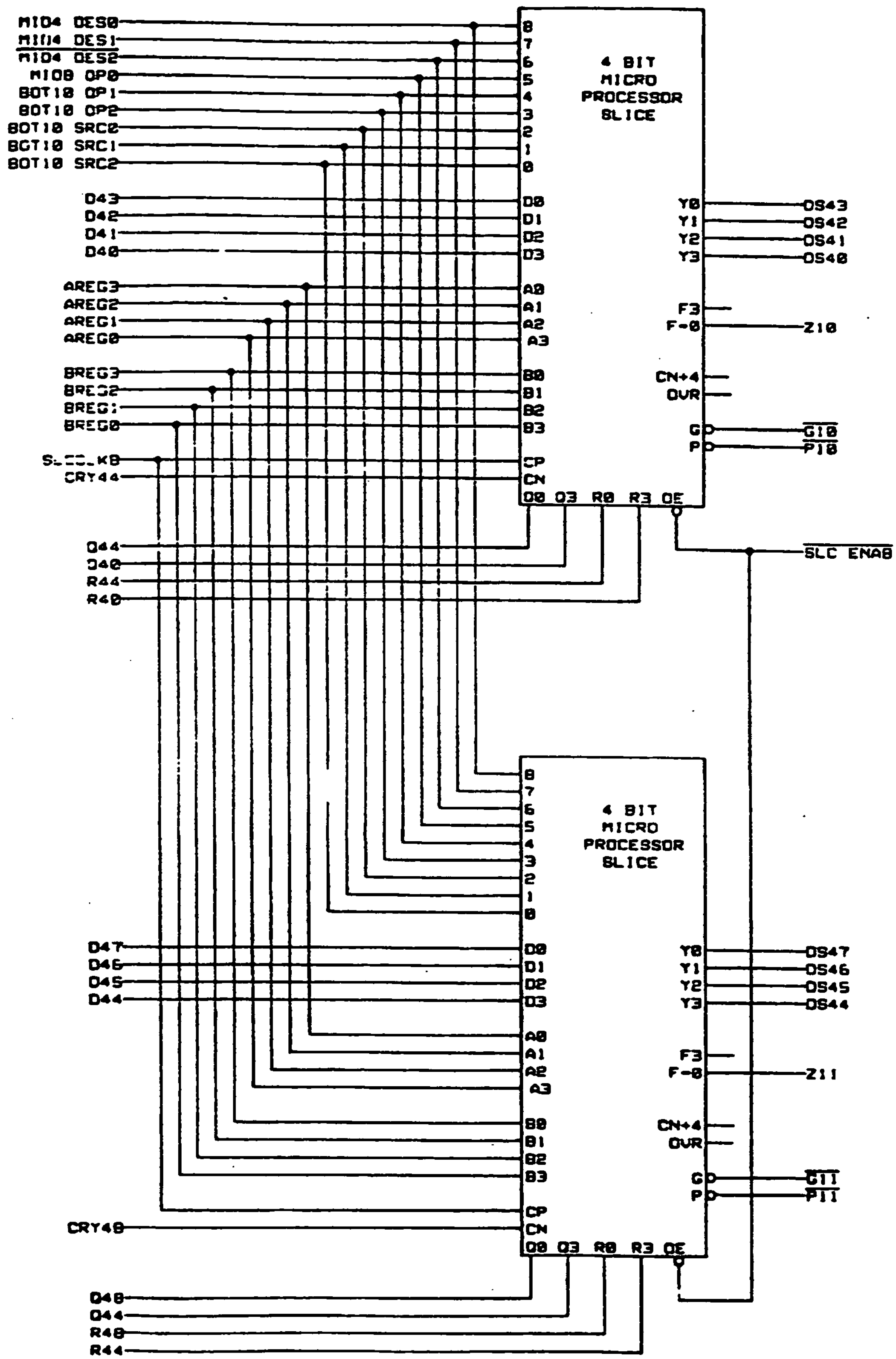

TOP10BOT2 DES0
TOP10BOT2 DES1
TOP10BOT2 DES2
MID8 OP0
BOT10 OP1
BOT10 OP2
BOT10 SRC0
BOT10 SRC1
BOT10 SRC2
D35
D34
D33
D32
AREG3
AREG2
AREG1
AREG0
BREG3
BREG2
BREG1
BREG0
SLCCLKB
CRY36
4 BIT MICRO PROCESSOR SLICE
DS35
DS34
DS33
DS32
Z8
G8
P8
SLC ENAB
D36
D32
R36
R32
D39
D38
D37
D36
DS39
DS38
DS37
DS36
Z9
G9
P9
CRY40
D40
D36
R40
R36

FIG. 9



MID4 DES0
MID4 DES1
MID4 DES2
MID8 OP0
BOT10 OP1
BOT10 OP2
BOT10 SRC2
BOT10 SRC1
BOT10 SRC2
D43
D42
D41
D40
AREG3
AREG2
AREG1
AREG0
BREG3
BREG2
BREG1
BREG0
CRY44
4 BIT
MICRO
PROCESSOR
SLICE
DS43
DS42
DS41
DS40
Z10
G10
P10
SLC ENAB
Q44
Q40
R44
R40
D47
D46
D45
D44
DS47
DS46
DS45
DS44
Z11
G11
P11
CRY48
Q48
Q44
R48
R44

FIG. 10

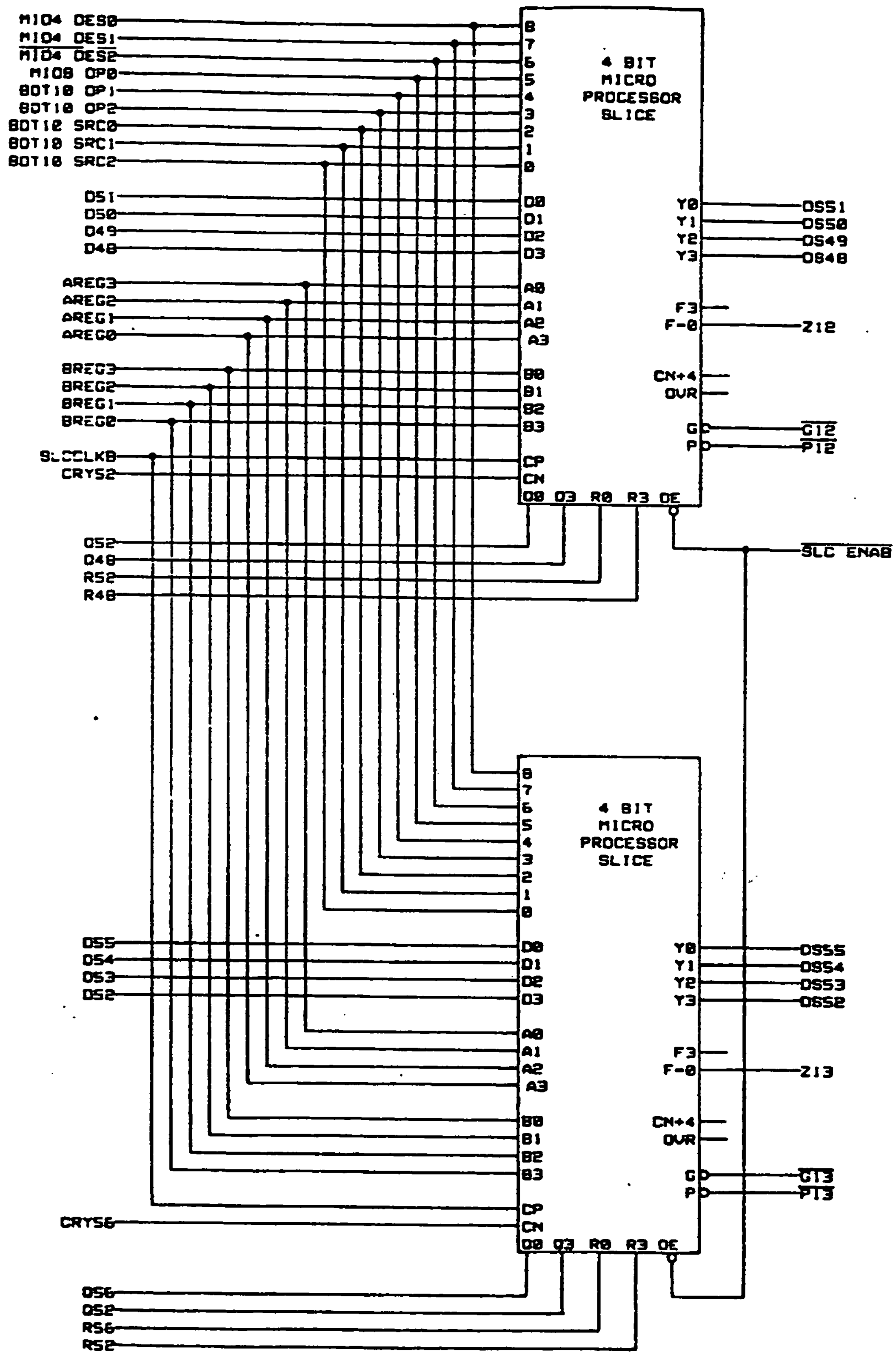

MID4 DES0
MID4 DES1
MID4 DES2
MID8 OP0
BOT10 OP1
BOT10 OP2
BOT10 SRC0
BOT10 SRC1
BOT10 SRC2
DS1
DS0
D49
D48
AREG3
AREG2
AREG1
AREG0
BREG3
BREG2
BREG1
BREG0
SLCCLKB
CRYS2
DS2
D48
RS2
R48
4 BIT MICRO PROCESSOR SLICE
8
7
6
5
4
3
2
1
0
D0
D1
D2
D3
A0
A1
A2
A3
B0
B1
B2
B3
CP
CN
Q0 Q3 R0 R3 OE
Y0
Y1
Y2
Y3
F3
F=0
CN+4
OVR
G
P
DSS1
DSS0
DS49
DS48
Z12
G12
P12
SLC ENAB
4 BIT MICRO PROCESSOR SLICE
DS5
DS4
DS3
DS2
CRYS6
DS6
DS2
RS6
RS2
DSS5
DSS4
DSS3
DSS2
Z13
G13
P13

FIG. 11

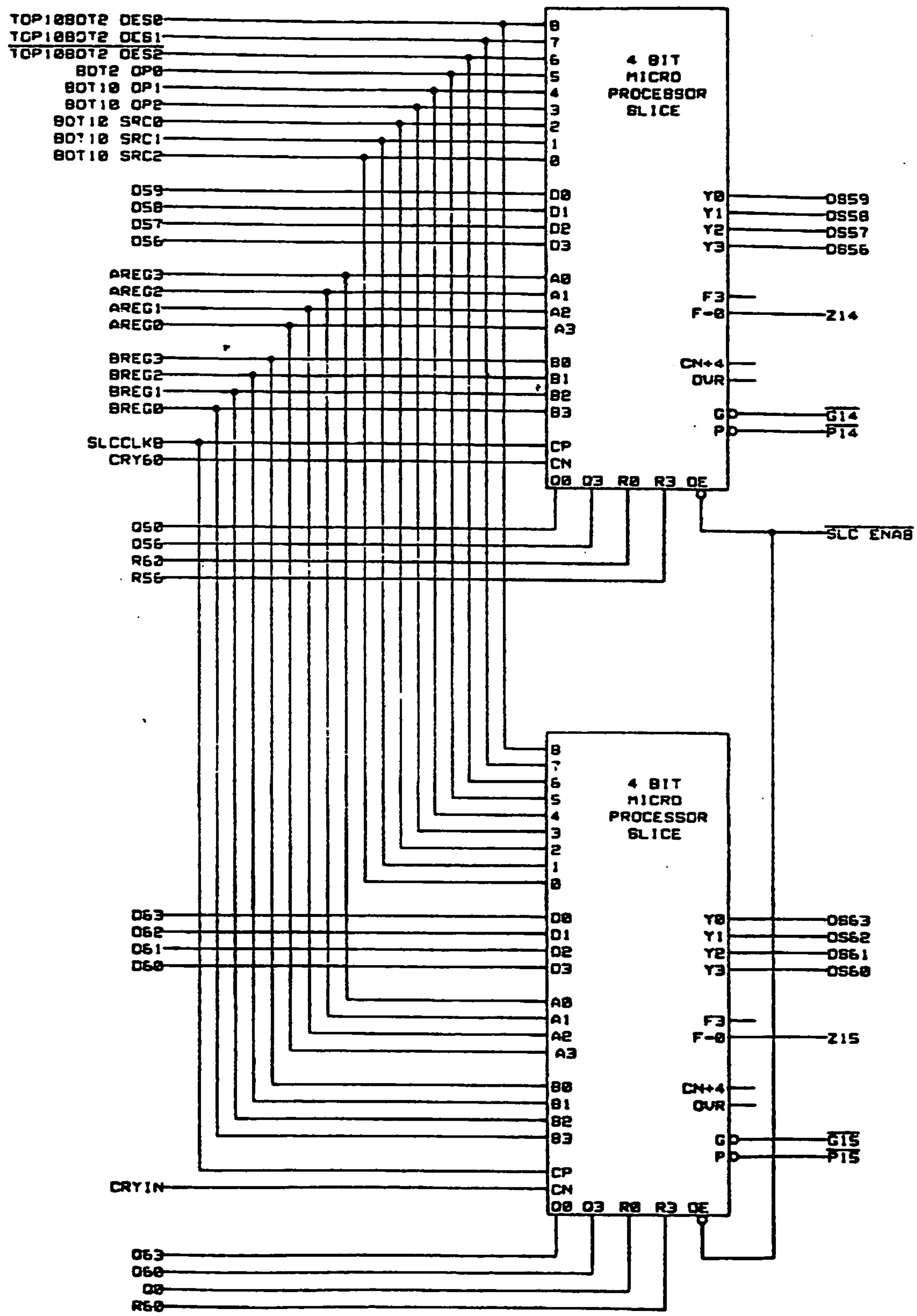

TOP108OT2 DES0
TOP108OT2 DES1
TOP108OT2 DES2
BOT2 OP0
BOT10 OP1
BOT10 OP2
BOT10 SRC0
BOT10 SRC1
BOT10 SRC2
DS9
DS8
DS7
DS6
AREG3
AREG2
AREG1
AREG0
BREG3
BREG2
BREG1
BREG0
SLCCLKB
CRY60
DS0
DS6
R62
R56
4 BIT MICRO PROCESSOR SLICE
Y0
Y1
Y2
Y3
DS59
DS58
DS57
DS56
F3
F-0
Z14
CN+4
OVR
G
P
G14
P14
CP
CN
Q0 Q3 R0 R3 OE
SLC ENAB
4 BIT MICRO PROCESSOR SLICE
D63
D62
D61
D60
DS63
DS62
DS61
DS60
Z15
G15
P15
CRYIN
D63
D60
Q0
R60

FIG. 12

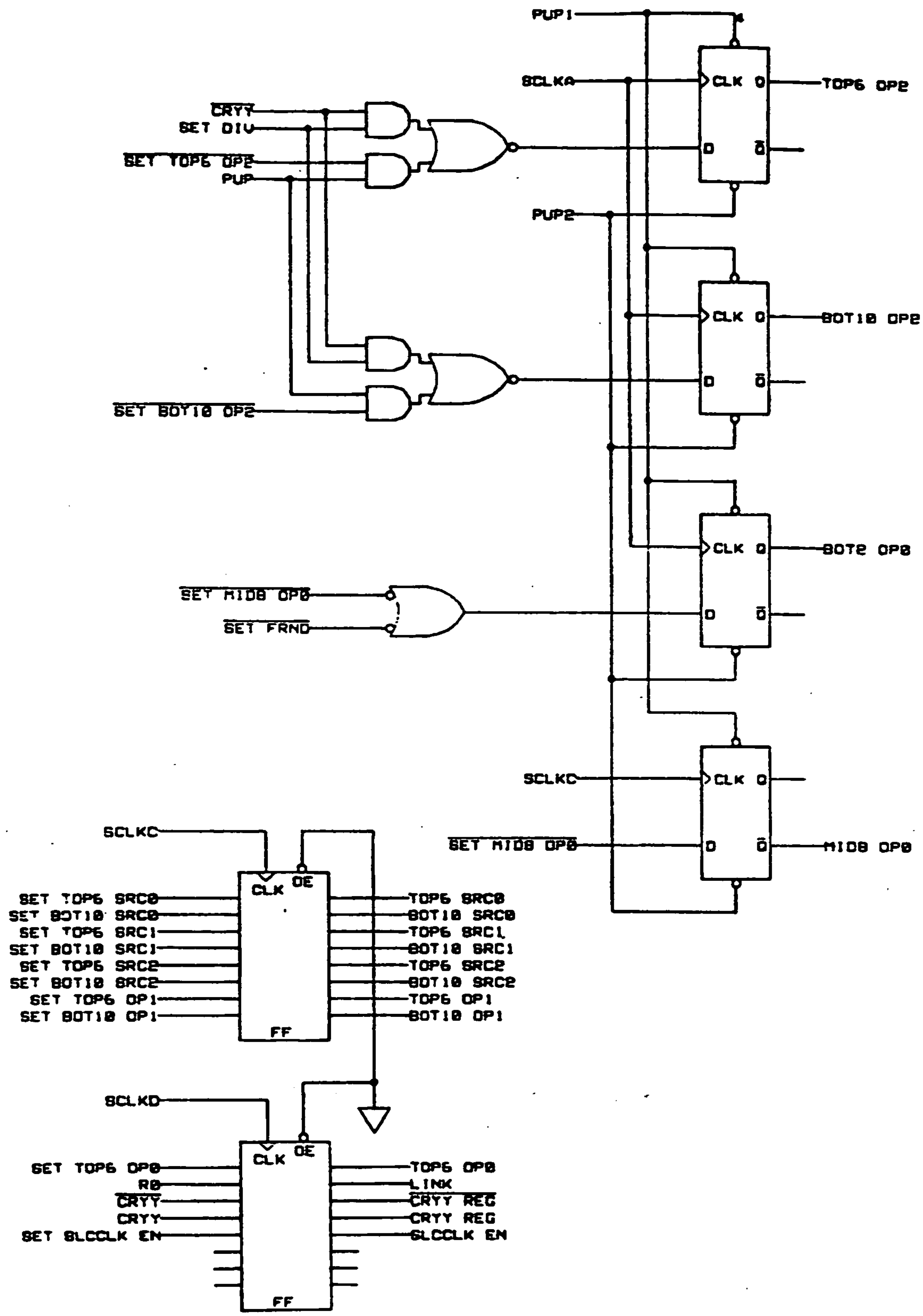

PUP1
SCLKA
CRYY
SET DIV
SET TOP6 OP2
PUP
CLK Q
D Q̄
TOP6 OP2
PUP2
BOT10 OP2
SET BOT10 OP2
BOT2 OP0
SET MID8 OP0
SET FRND
SCLKC
SET MID8 OP0
MID8 OP0
SCLKC
CLK
OE
SET TOP6 SRC0
SET BOT10 SRC0
SET TOP6 SRC1
SET BOT10 SRC1
SET TOP6 SRC2
SET BOT10 SRC2
SET TOP6 OP1
SET BOT10 OP1
FF
TOP6 SRC0
BOT10 SRC0
TOP6 SRC1
BOT10 SRC1
TOP6 SRC2
BOT10 SRC2
TOP6 OP1
BOT10 OP1
SCLKD
SET TOP6 OP0
R0
CRYY
CRYY
SET SLCCLK EN
TOP6 OP0
LINK
CRYY REG
CRYY REG
SLCCLK EN

FIG. 13

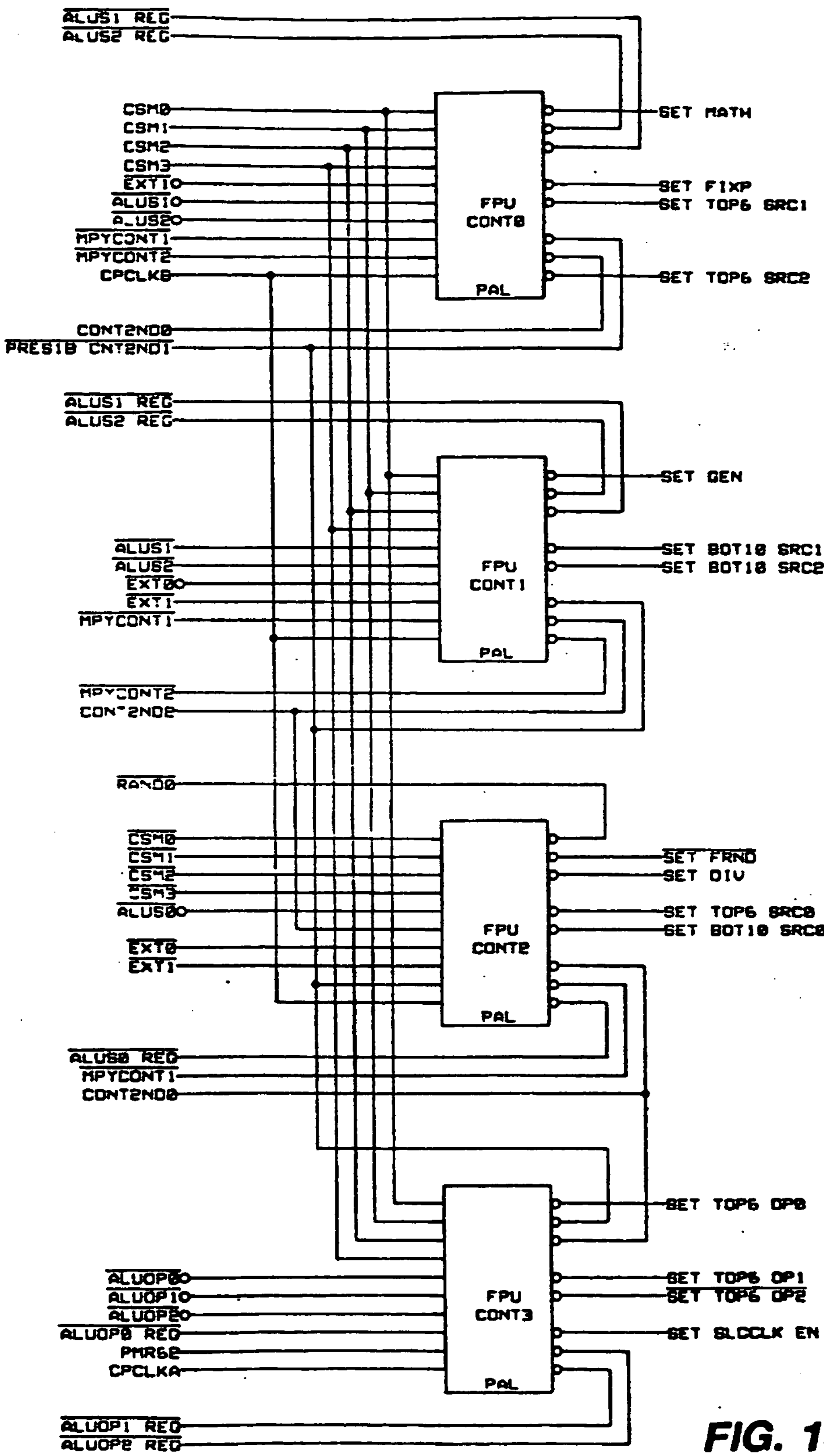

ALUS1 REG
ALUS2 REG
CSM0
CSM1
CSM2
CSM3
EXT1
ALUS1
ALUS2
MPYCONT1
MPYCONT2
CPCLKB
CONT2ND0
PRES1B CNT2ND1
FPU CONT0
PAL
SET MATH
SET FIXP
SET TOP6 SRC1
SET TOP6 SRC2
ALUS1 REG
ALUS2 REG
ALUS1
ALUS2
EXT0
EXT1
MPYCONT1
MPYCONT2
CONT2ND2
FPU CONT1
PAL
SET GEN
SET BOT10 SRC1
SET BOT10 SRC2
RAND0
CSM0
CSM1
CSM2
CSM3
ALUS0
EXT0
EXT1
FPU CONT2
PAL
SET FRND
SET DIV
SET TOP6 SRC0
SET BOT10 SRC0
ALUS0 REG
MPYCONT1
CONT2ND0
FPU CONT3
PAL
SET TOP6 OP0
SET TOP6 OP1
SET TOP6 OP2
SET SLCCLK EN
ALUOP0
ALUOP1
ALUOP2
ALUOP0 REG
PMR62
CPCLKA
ALUOP1 REG
ALUOP2 REG

FIG. 14

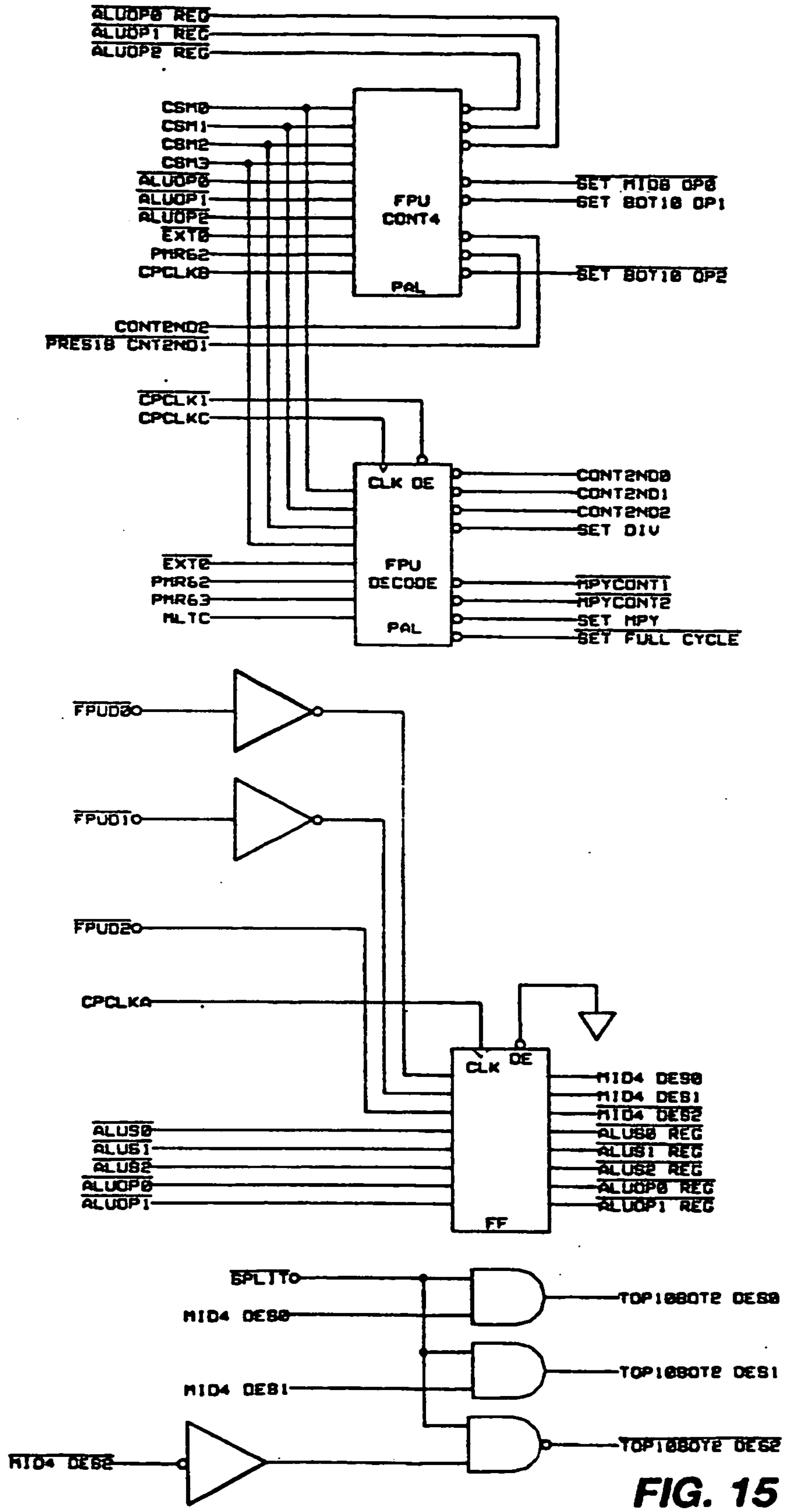
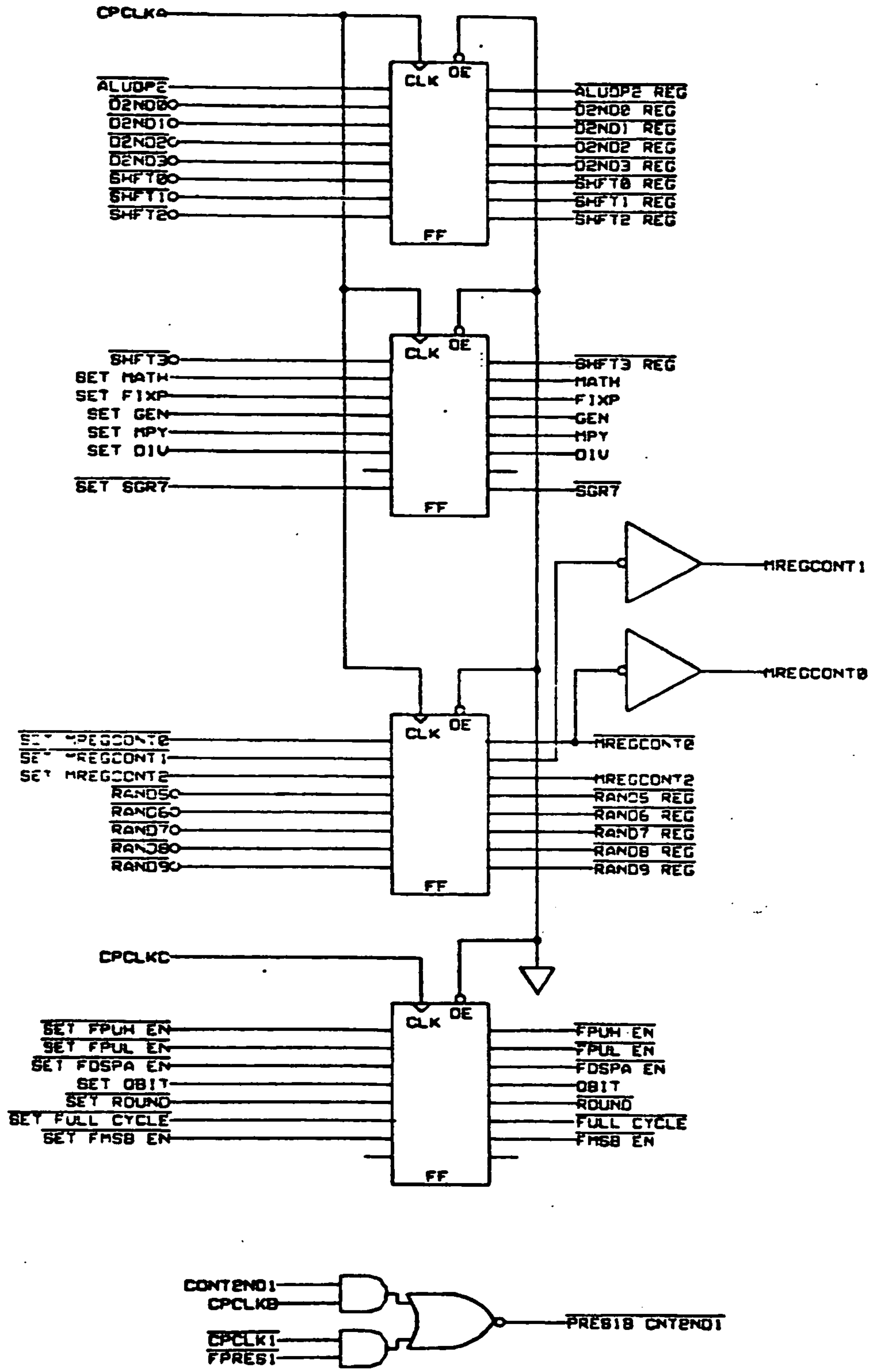

ALUOP0 REG
ALUOP1 REG
ALUOP2 REG
CSM0
CSM1
CSM2
CSM3
ALUOP0
ALUOP1
ALUOP2
EXT0
PMR62
CPCLKB
FPU
CONT4
PAL
SET MID8 OP0
SET BOT10 OP1
SET BOT10 OP2
CONT2ND2
PRES10 CNT2ND1
CPCLKI
CPCLKC
CLK OE
CONT2ND0
CONT2ND1
CONT2ND2
SET DIV
EXT0
PMR62
PMR63
MLTC
FPU
DECODE
PAL
MPYCONT1
MPYCONT2
SET MPY
SET FULL CYCLE
FPUD0
FPUD1
FPUD2
CPCLKA
CLK
OE
FF
MID4 DES0
MID4 DES1
MID4 DES2
ALUS0 REG
ALUS1 REG
ALUS2 REG
ALUOP0 REG
ALUOP1 REG
ALUS0
ALUS1
ALUS2
ALUOP0
ALUOP1
SPLIT
MID4 DES0
MID4 DES1
MID4 DES2
TOP10BOT2 DES0
TOP10BOT2 DES1
TOP10BOT2 DES2

FIG. 15



CPCLKA
ALUOP2
O2NDO0
O2NDO1
O2NDO2
O2NDO3
SHFT0
SHFT1
SHFT2
CLK
OE
FF
ALUOP2 REG
O2ND0 REG
O2ND1 REG
O2ND2 REG
O2ND3 REG
SHFT0 REG
SHFT1 REG
SHFT2 REG
SHFT3
SET MATH
SET FIXP
SET GEN
SET MPY
SET DIV
SET SGR7
SHFT3 REG
MATH
FIXP
GEN
MPY
DIV
SGR7
MREGCONT1
MREGCONT0
SET MREGCONT0
SET MREGCONT1
SET MREGCONT2
RAND5
RAND6
RAND7
RAND8
RAND9
MREGCONT0
MREGCONT2
RAND5 REG
RAND6 REG
RAND7 REG
RAND8 REG
RAND9 REG
CPCLKC
SET FPUH EN
SET FPUL EN
SET FDSPA EN
SET OBIT
SET ROUND
SET FULL CYCLE
SET FMSB EN
FPUH EN
FPUL EN
FDSPA EN
OBIT
ROUND
FULL CYCLE
FMSB EN
CONT2ND1
CPCLKB
CPCLK1
FPRES1
PRES18 CNT2ND1

FIG. 16

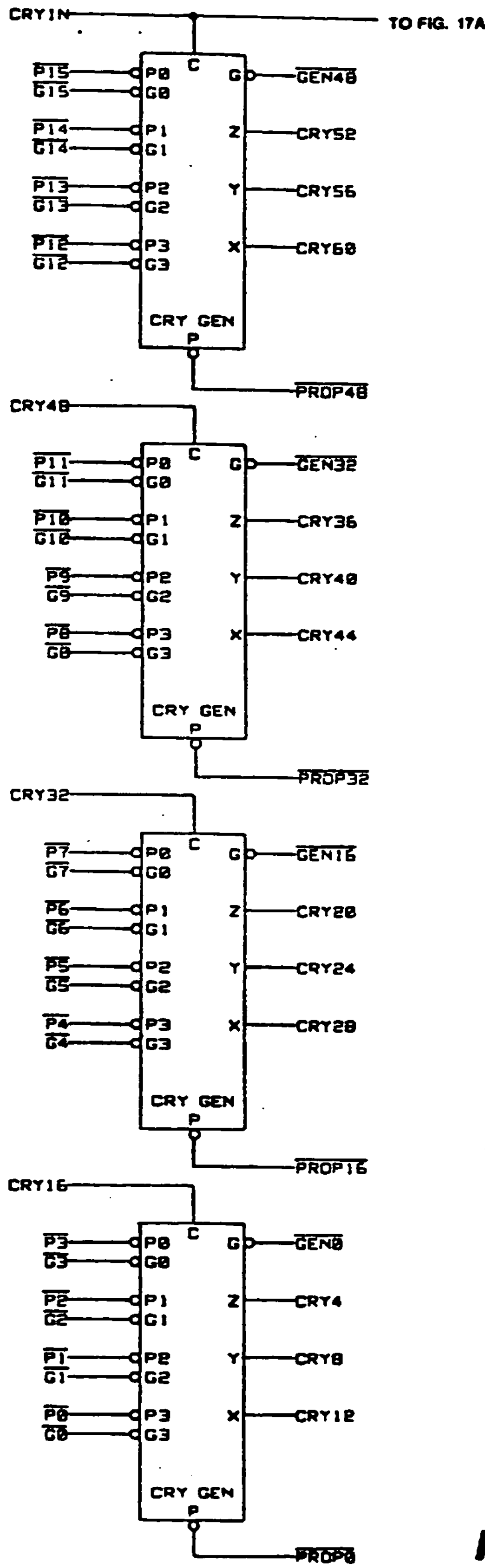

CRYIN
TO FIG. 17A
P15
G15
P14
G14
P13
G13
P12
G12
P0
G0
P1
G1
P2
G2
P3
G3
C
G
Z
Y
X
GEN48
CRY52
CRY56
CRY60
CRY GEN
P
PROP48
CRY48
P11
G11
P10
G10
P9
G9
P8
G8
GEN32
CRY36
CRY40
CRY44
PROP32
CRY32
P7
G7
P6
G6
P5
G5
P4
G4
GEN16
CRY20
CRY24
CRY28
PROP16
CRY16
P3
G3
P2
G2
P1
G1
P0
G0
GEN0
CRY4
CRY8
CRY12
PROP0

FIG. 17

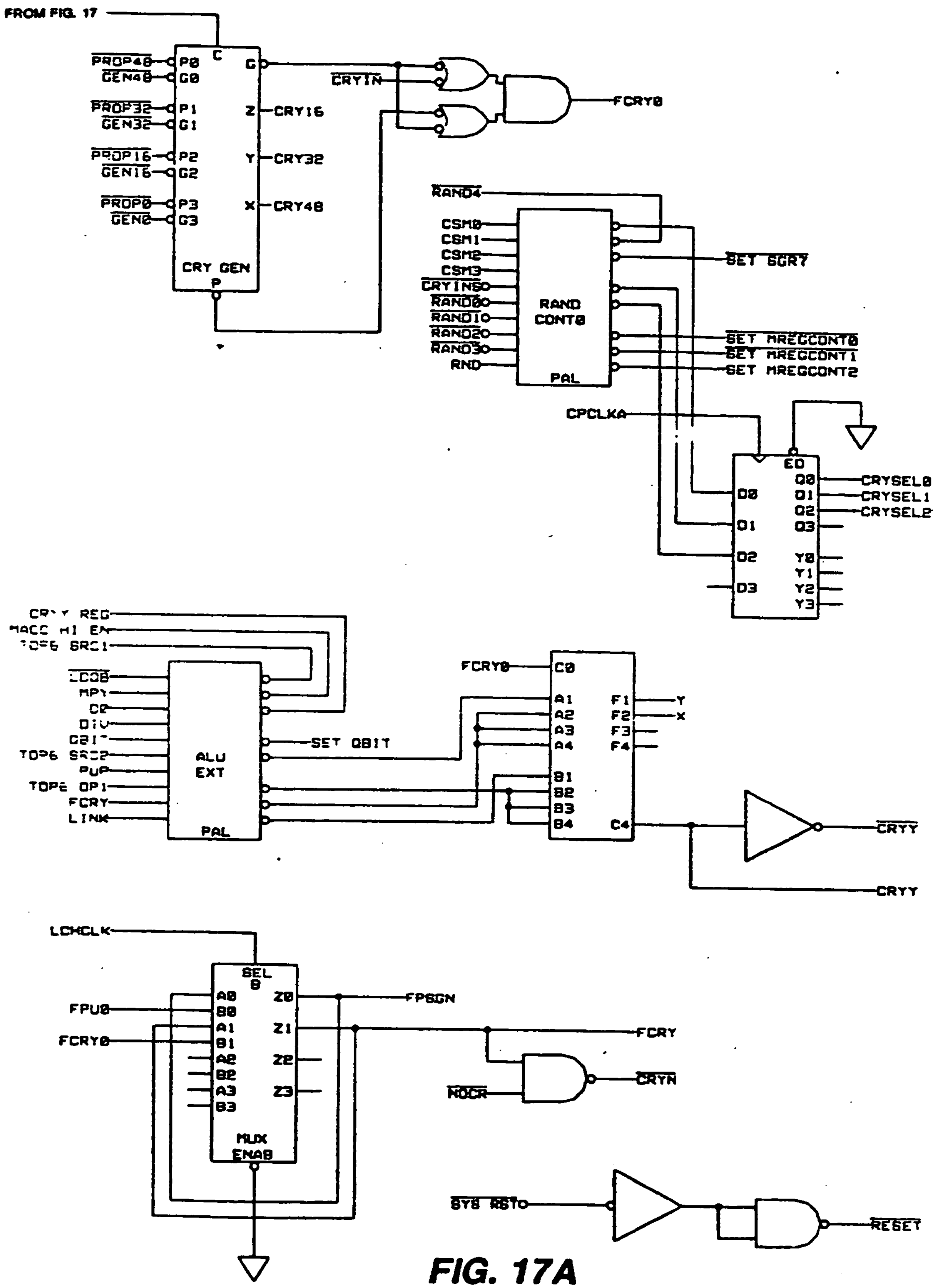

FROM FIG. 17
PROP48
GEN48
PROP32
GEN32
PROP16
GEN16
PROP0
GEN0
P0
G0
P1
G1
P2
G2
P3
G3
C
G
Z
Y
X
P
CRY GEN
CRY16
CRY32
CRY48
CRYIN
FCRY0
RAND4
CSM0
CSM1
CSM2
CSM3
CRYINS0
RAND0
RAND1
RAND2
RAND3
RND
RAND
CONT0
PAL
SET SGR7
SET MREGCONT0
SET MREGCONT1
SET MREGCONT2
CPCLKA
EO
D0
D1
D2
D3
Q0
Q1
Q2
Q3
Y0
Y1
Y2
Y3
CRYSEL0
CRYSEL1
CRYSEL2
MACC HI EN
LCOB
MPY
DIV
TOP6 SRC2
TOP6 OP1
FCRY
LINK
ALU
EXT
PAL
SET QBIT
FCRY0
C0
A1
A2
A3
A4
B1
B2
B3
B4
F1
F2
F3
F4
C4
Y
X
CRYY
CRYY
LCHCLK
SEL
B
A0
B0
A1
B1
A2
B2
A3
B3
Z0
Z1
Z2
Z3
MUX
ENAB
FPU0
FCRY0
FPSGN
FCRY
NOCR
CRYN
SYS RST
RESET

FIG. 17A

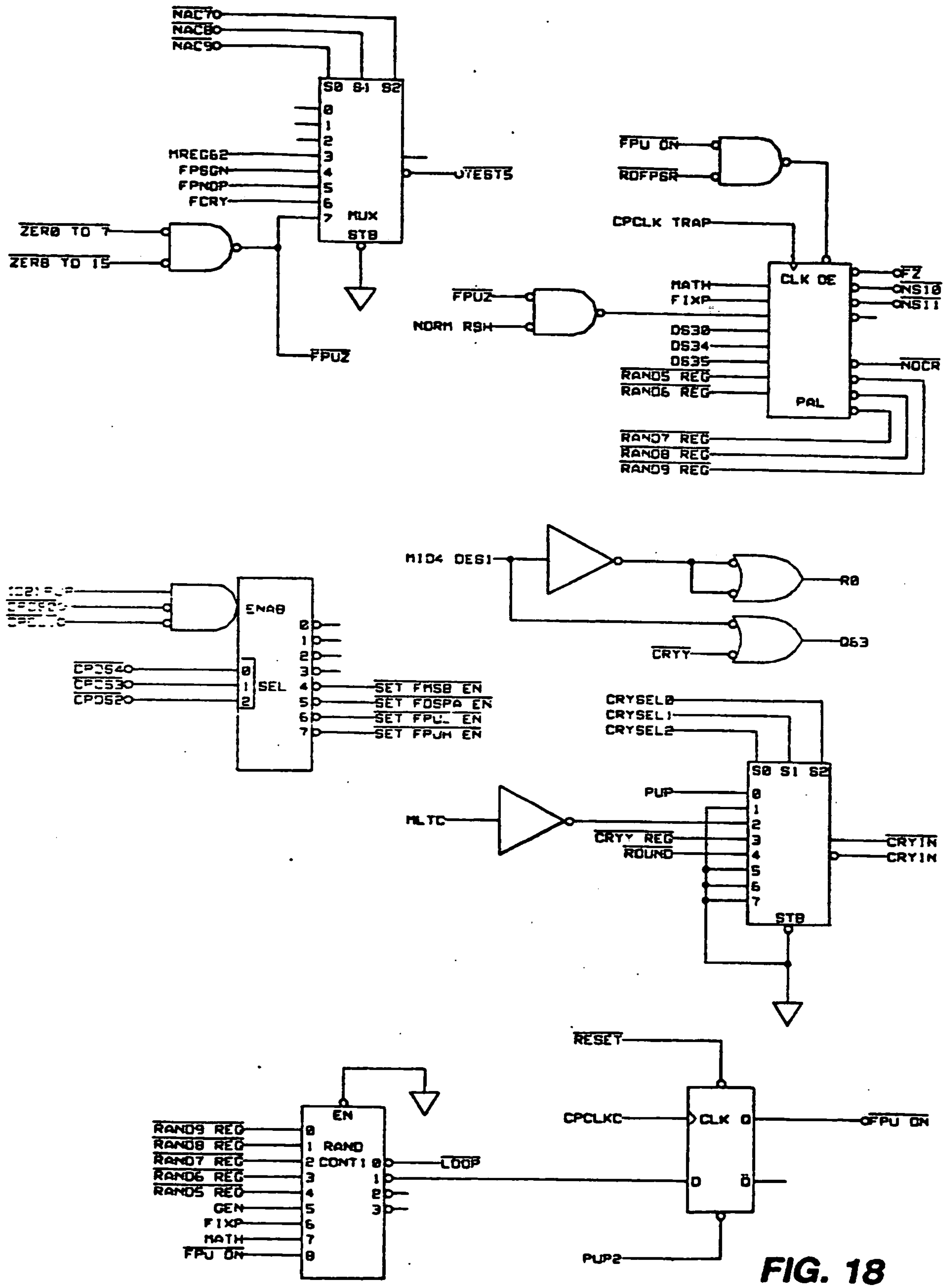
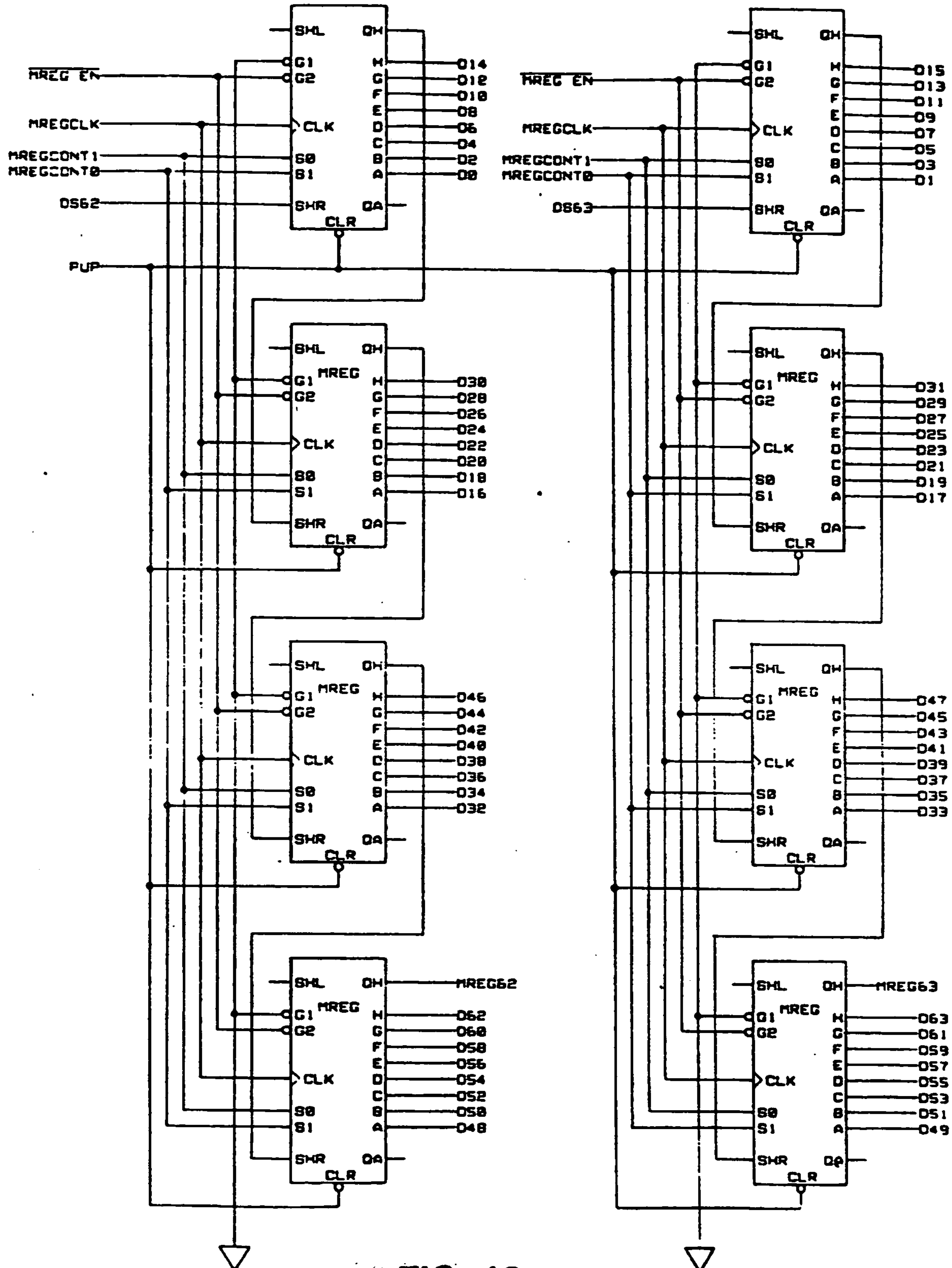

NAC70
NAC80
NAC90
S0 S1 S2
MREG62
FPSGN
FPNOP
FCRY
MUX
STB
UTEST5
ZER0 TO 7
ZER8 TO 15
FPUZ
FPU ON
RDFP6R
CPCLK TRAP
CLK OE
FZ
ONS10
ONS11
MATH
FIXP
FPUZ
NORM RSH
DS30
DS34
DS35
RAND5 REG
RAND6 REG
NOCR
PAL
RAND7 REG
RAND8 REG
RAND9 REG
MID4 DES1
R0
DS3
CRYY
ENAB
SEL
SET FMSB EN
SET FDSPA EN
SET FPUL EN
SET FPUH EN
CRYSEL0
CRYSEL1
CRYSEL2
PUP
MLTC
CRYY REG
ROUND
CRYIN
RESET
CPCLKC
CLK Q
FPU ON
PUP2
EN
RAND9 REG
RAND8 REG
RAND7 REG
RAND6 REG
RAND5 REG
GEN
FIXP
MATH
FPU ON
RAND CONT1
LOOP

FIG. 18



MREG EN
MREGCLK
MREGCONT1
MREGCONT0
DS62
PUP
SHL
QH
G1
G2
CLK
S0
S1
SHR
QA
CLR
H
G
F
E
D
C
B
A
MREG
D14
D12
D10
D8
D6
D4
D2
D0
D30
D28
D26
D24
D22
D20
D18
D16
D46
D44
D42
D40
D38
D36
D34
D32
MREG62
D62
D60
D58
D56
D54
D52
D50
D48
DS63
D15
D13
D11
D9
D7
D5
D3
D1
D31
D29
D27
D25
D23
D21
D19
D17
D47
D45
D43
D41
D39
D37
D35
D33
MREG63
D63
D61
D59
D57
D55
D53
D51
D49

FIG. 19

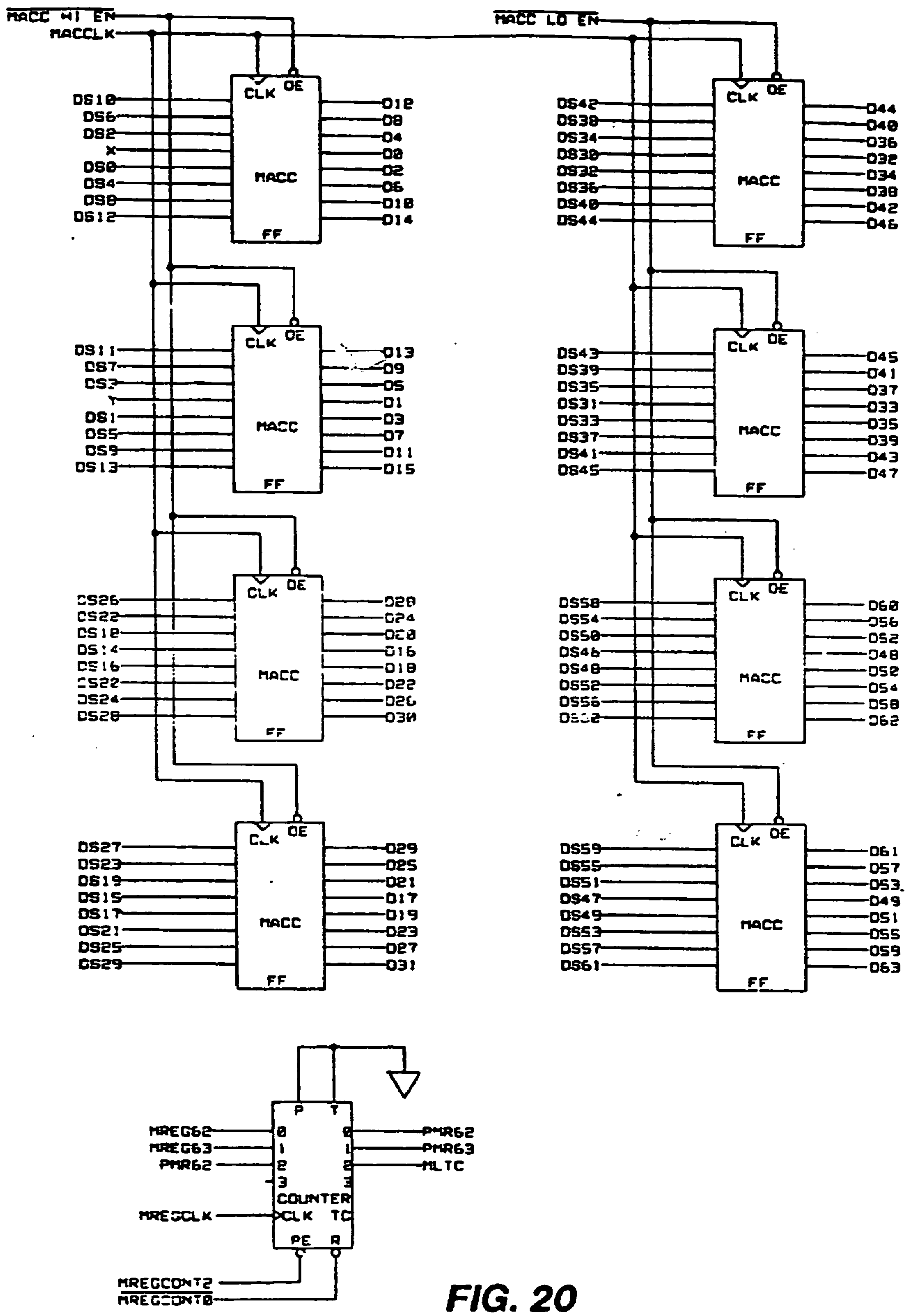

MACC HI EN
MACCLK
MACC LO EN
CLK
OE
MACC
FF
COUNTER
MREG62
MREG63
PMR62
PMR63
MLTC
MREGCLK
MREGCONT2
MREGCONT0

FIG. 20

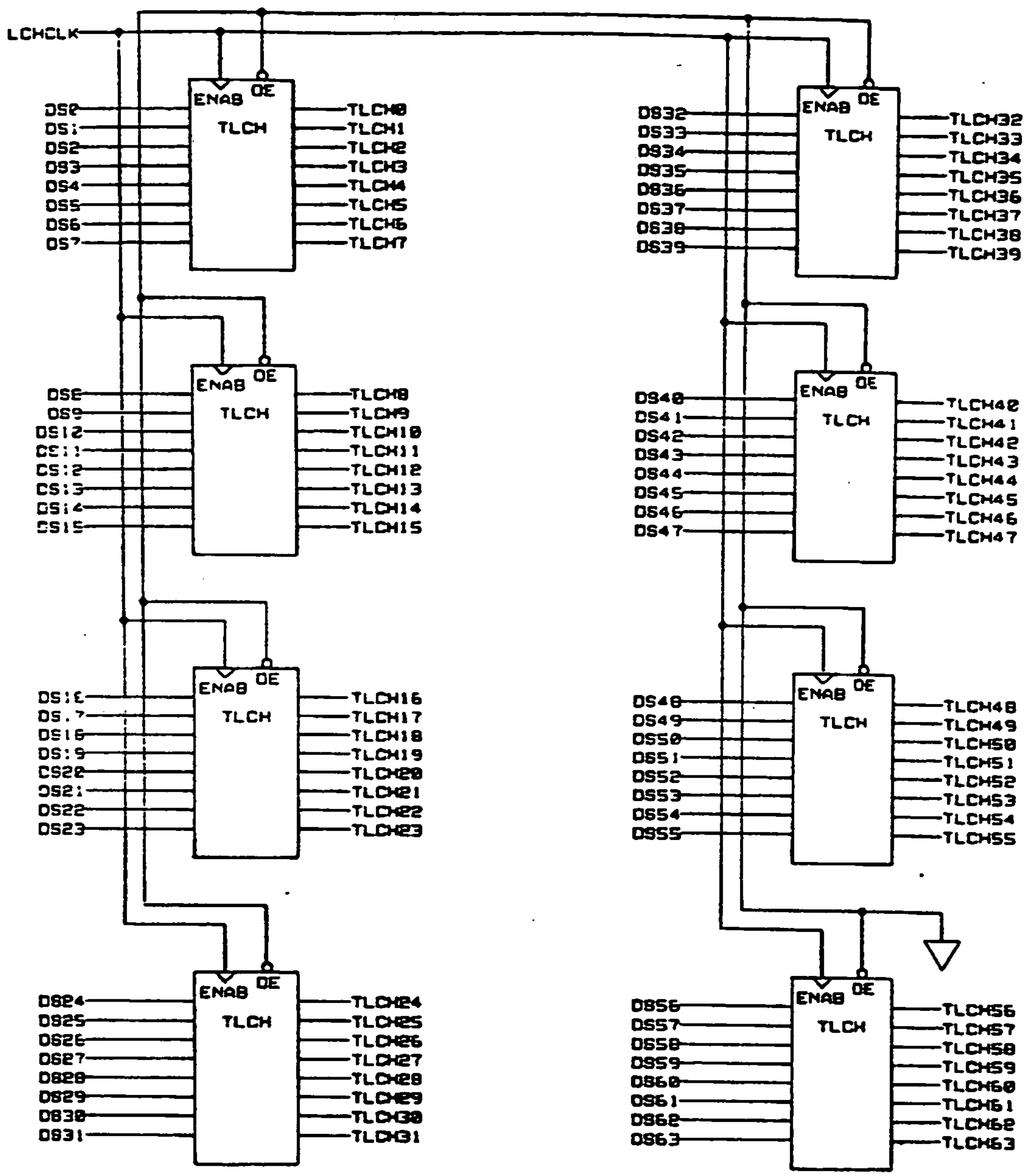

LCHCLK
ENAB OE
TLCH
DS0 DS1 DS2 DS3 DS4 DS5 DS6 DS7
TLCH0 TLCH1 TLCH2 TLCH3 TLCH4 TLCH5 TLCH6 TLCH7
DS8 DS9 DS10 DS11 DS12 DS13 DS14 DS15
TLCH8 TLCH9 TLCH10 TLCH11 TLCH12 TLCH13 TLCH14 TLCH15
DS16 DS17 DS18 DS19 DS20 DS21 DS22 DS23
TLCH16 TLCH17 TLCH18 TLCH19 TLCH20 TLCH21 TLCH22 TLCH23
DS24 DS25 DS26 DS27 DS28 DS29 DS30 DS31
TLCH24 TLCH25 TLCH26 TLCH27 TLCH28 TLCH29 TLCH30 TLCH31
DS32 DS33 DS34 DS35 DS36 DS37 DS38 DS39
TLCH32 TLCH33 TLCH34 TLCH35 TLCH36 TLCH37 TLCH38 TLCH39
DS40 DS41 DS42 DS43 DS44 DS45 DS46 DS47
TLCH40 TLCH41 TLCH42 TLCH43 TLCH44 TLCH45 TLCH46 TLCH47
DS48 DS49 DS50 DS51 DS52 DS53 DS54 DS55
TLCH48 TLCH49 TLCH50 TLCH51 TLCH52 TLCH53 TLCH54 TLCH55
DS56 DS57 DS58 DS59 DS60 DS61 DS62 DS63
TLCH56 TLCH57 TLCH58 TLCH59 TLCH60 TLCH61 TLCH62 TLCH63

FIG. 21

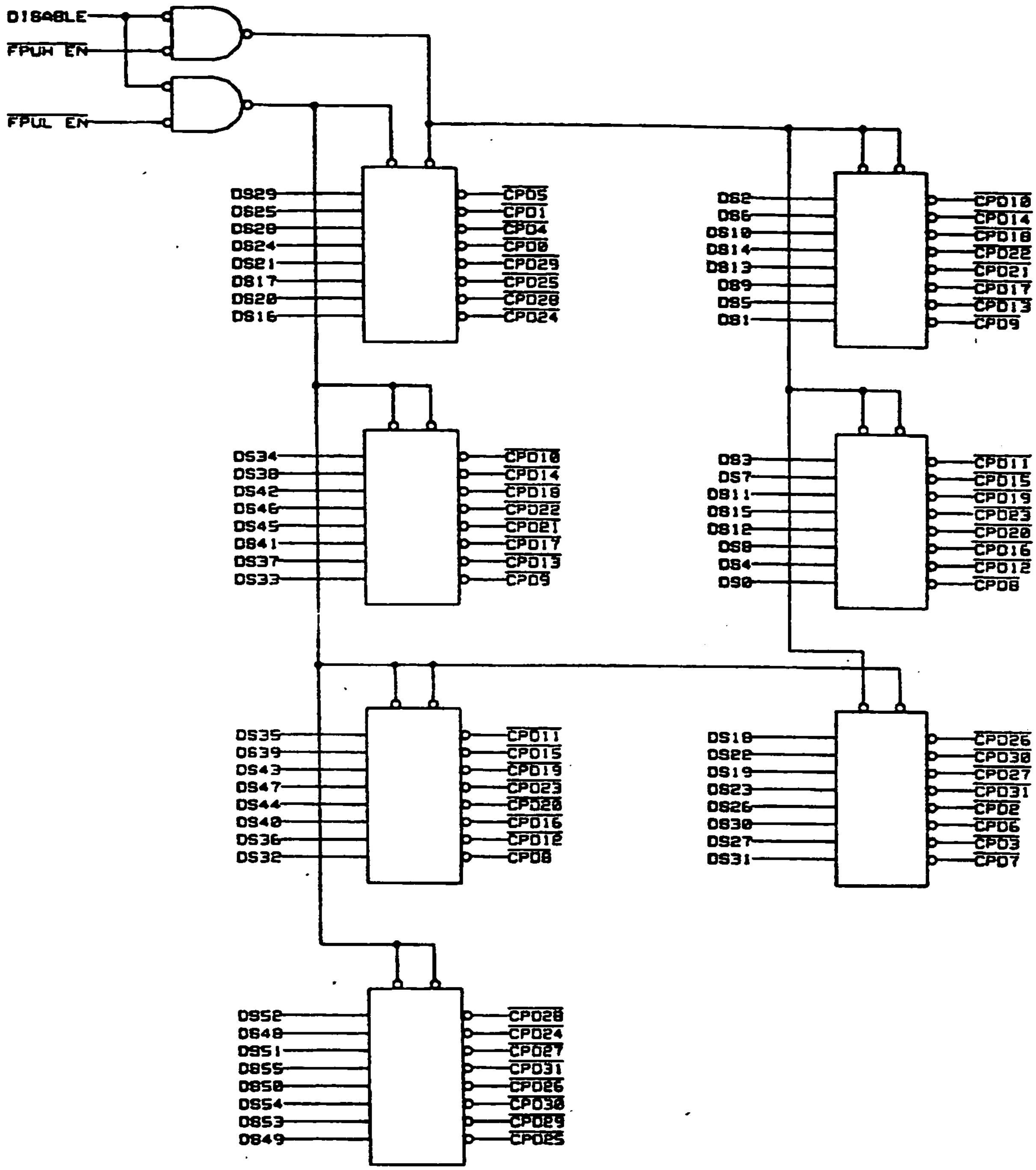

DISABLE
FPUH EN
FPUL EN
DS29
DS25
DS28
DS24
DS21
DS17
DS20
DS16
CPD5
CPD1
CPD4
CPD0
CPD29
CPD25
CPD28
CPD24
DS2
DS6
DS10
DS14
DS13
DS9
DS5
DS1
CPD10
CPD14
CPD18
CPD22
CPD21
CPD17
CPD13
CPD9
DS34
DS38
DS42
DS46
DS45
DS41
DS37
DS33
CPD10
CPD14
CPD18
CPD22
CPD21
CPD17
CPD13
CPD9
DS3
DS7
DS11
DS15
DS12
DS8
DS4
DS0
CPD11
CPD15
CPD19
CPD23
CPD20
CPD16
CPD12
CPD8
DS35
DS39
DS43
DS47
DS44
DS40
DS36
DS32
CPD11
CPD15
CPD19
CPD23
CPD20
CPD16
CPD12
CPD8
DS18
DS22
DS19
DS23
DS26
DS30
DS27
DS31
CPD26
CPD30
CPD27
CPD31
CPD2
CPD6
CPD3
CPD7
DS52
DS48
DS51
DS55
DS50
DS54
DS53
DS49
CPD28
CPD24
CPD27
CPD31
CPD26
CPD30
CPD29
CPD25

FIG. 22

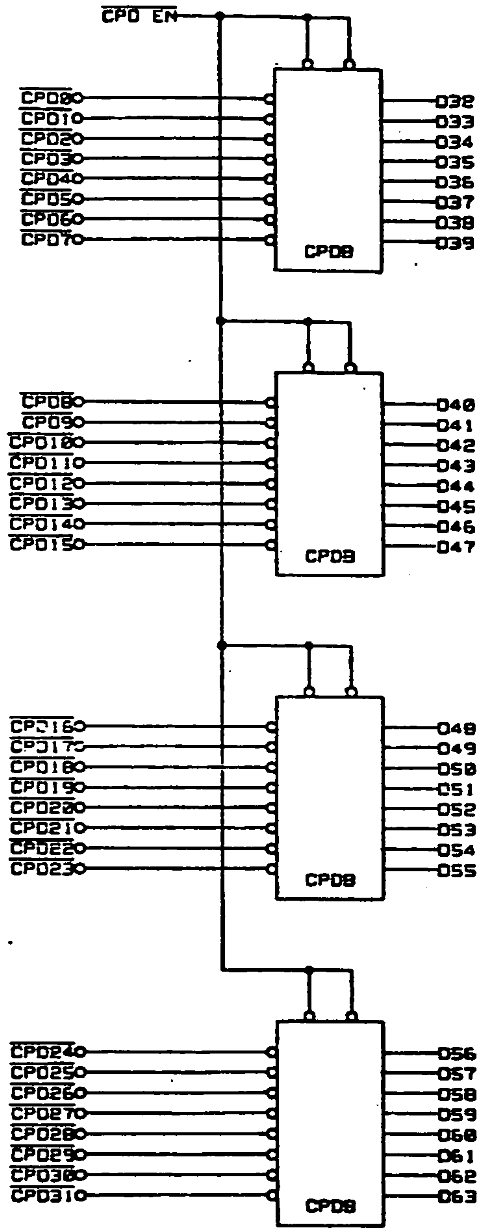

CPD EN
CPD0
CPD1
CPD2
CPD3
CPD4
CPD5
CPD6
CPD7
CPD8
D32
D33
D34
D35
D36
D37
D38
D39
CPD8
CPD9
CPD10
CPD11
CPD12
CPD13
CPD14
CPD15
CPD8
D40
D41
D42
D43
D44
D45
D46
D47
CPD16
CPD17
CPD18
CPD19
CPD20
CPD21
CPD22
CPD23
CPD8
D48
D49
D50
D51
D52
D53
D54
D55
CPD24
CPD25
CPD26
CPD27
CPD28
CPD29
CPD30
CPD31
CPD8
D56
D57
D58
D59
D60
D61
D62
D63

FIG. 23

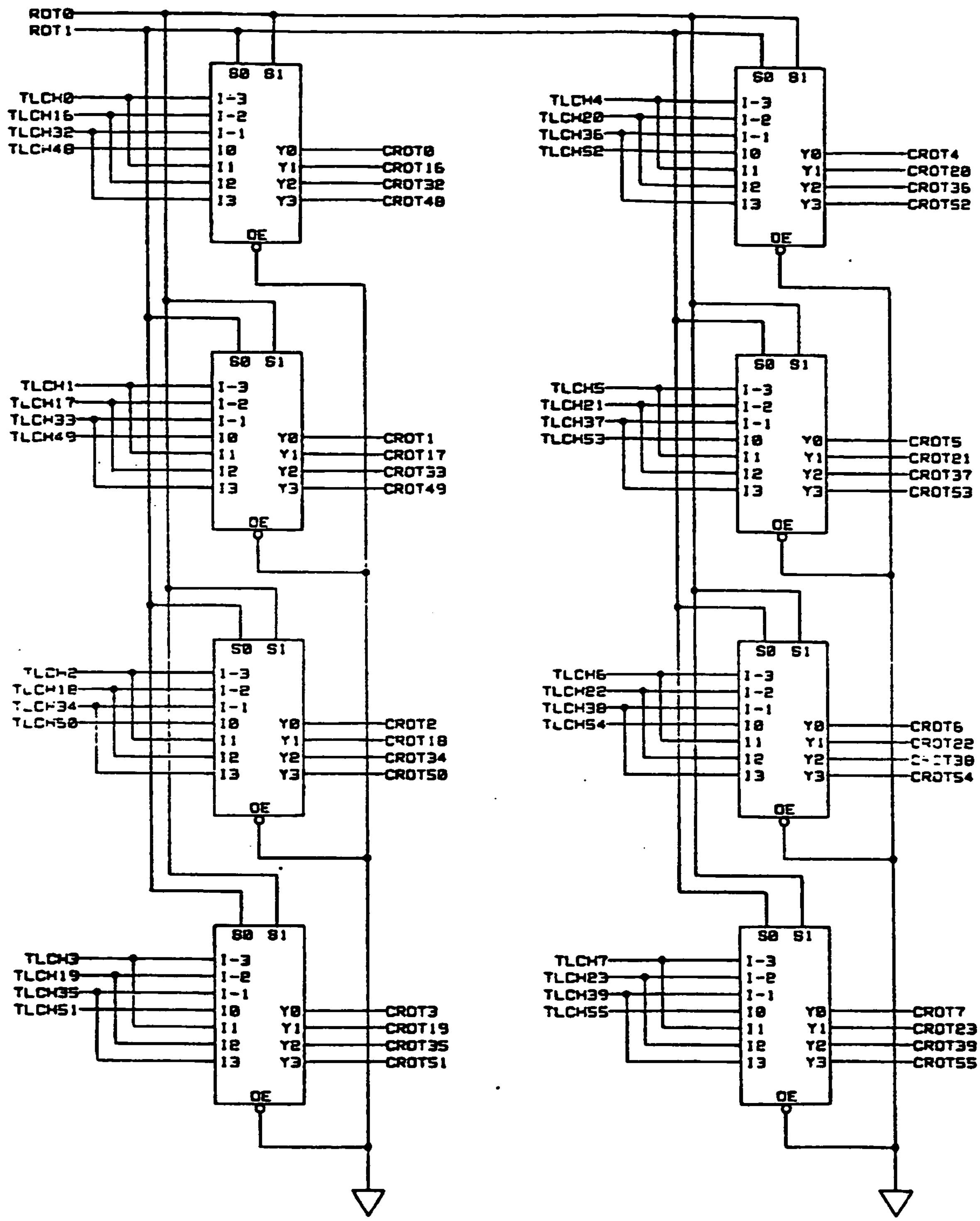

ROT0
ROT1
S0 S1
I-3
I-2
I-1
I0
I1
I2
I3
Y0
Y1
Y2
Y3
OE
TLCH0
TLCH16
TLCH32
TLCH48
CROT0
CROT16
CROT32
CROT48
TLCH1
TLCH17
TLCH33
TLCH49
CROT1
CROT17
CROT33
CROT49
TLCH2
TLCH18
TLCH34
TLCH50
CROT2
CROT18
CROT34
CROT50
TLCH3
TLCH19
TLCH35
TLCH51
CROT3
CROT19
CROT35
CROT51
TLCH4
TLCH20
TLCH36
TLCH52
CROT4
CROT20
CROT36
CROT52
TLCH5
TLCH21
TLCH37
TLCH53
CROT5
CROT21
CROT37
CROT53
TLCH6
TLCH22
TLCH38
TLCH54
CROT6
CROT22
CROT54
TLCH7
TLCH23
TLCH39
TLCH55
CROT7
CROT23
CROT39
CROT55

FIG. 24

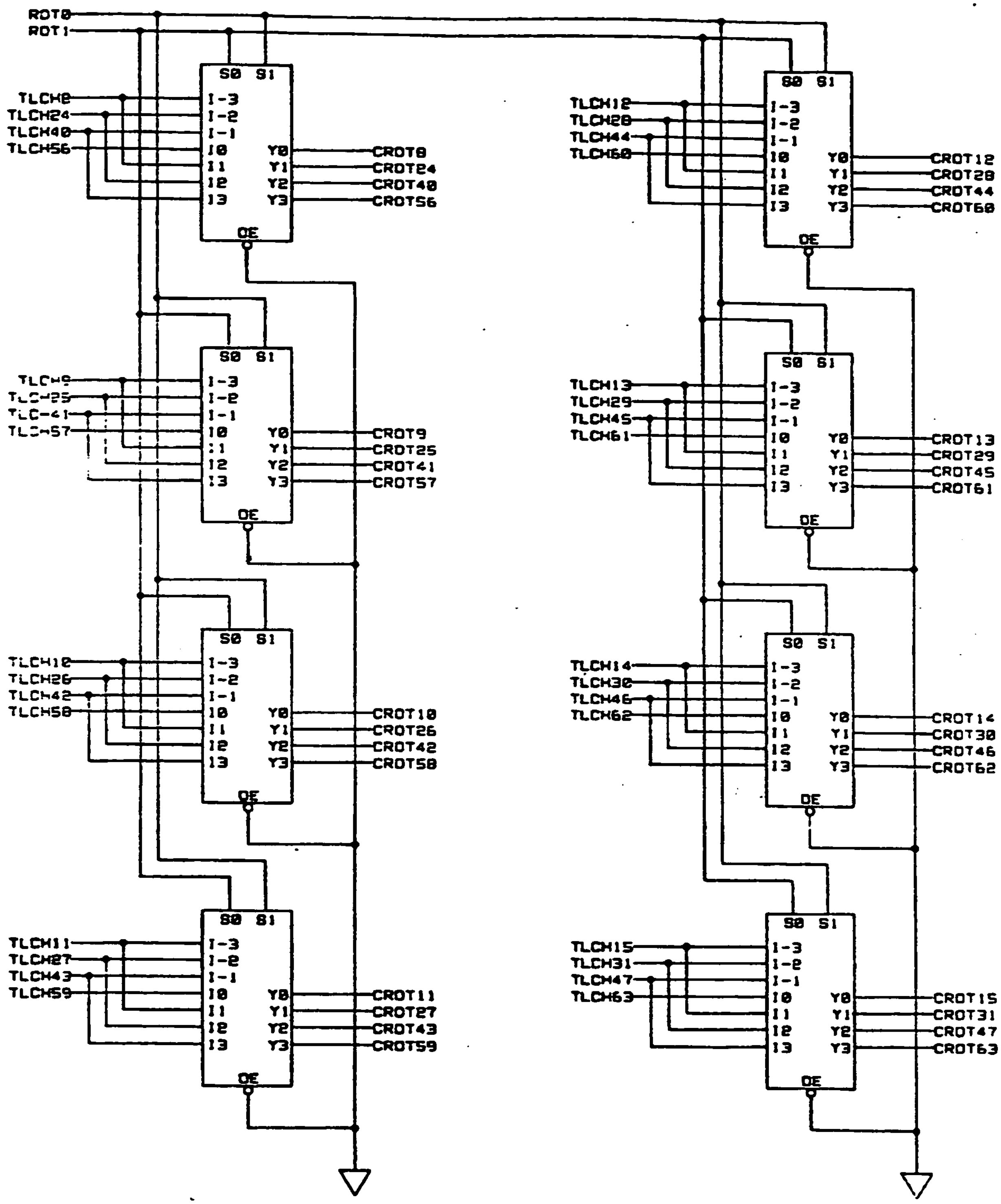

ROT0
ROT1
S0 S1
I-3
I-2
I-1
I0
I1
I2
I3
Y0
Y1
Y2
Y3
OE
TLCH8
TLCH24
TLCH40
TLCH56
CROT8
CROT24
CROT40
CROT56
TLCH9
TLCH25
TLCH41
TLCH57
CROT9
CROT25
CROT41
CROT57
TLCH10
TLCH26
TLCH42
TLCH58
CROT10
CROT26
CROT42
CROT58
TLCH11
TLCH27
TLCH43
TLCH59
CROT11
CROT27
CROT43
CROT59
TLCH12
TLCH28
TLCH44
TLCH60
CROT12
CROT28
CROT44
CROT60
TLCH13
TLCH29
TLCH45
TLCH61
CROT13
CROT29
CROT45
CROT61
TLCH14
TLCH30
TLCH46
TLCH62
CROT14
CROT30
CROT46
CROT62
TLCH15
TLCH31
TLCH47
TLCH63
CROT15
CROT31
CROT47
CROT63

FIG. 25

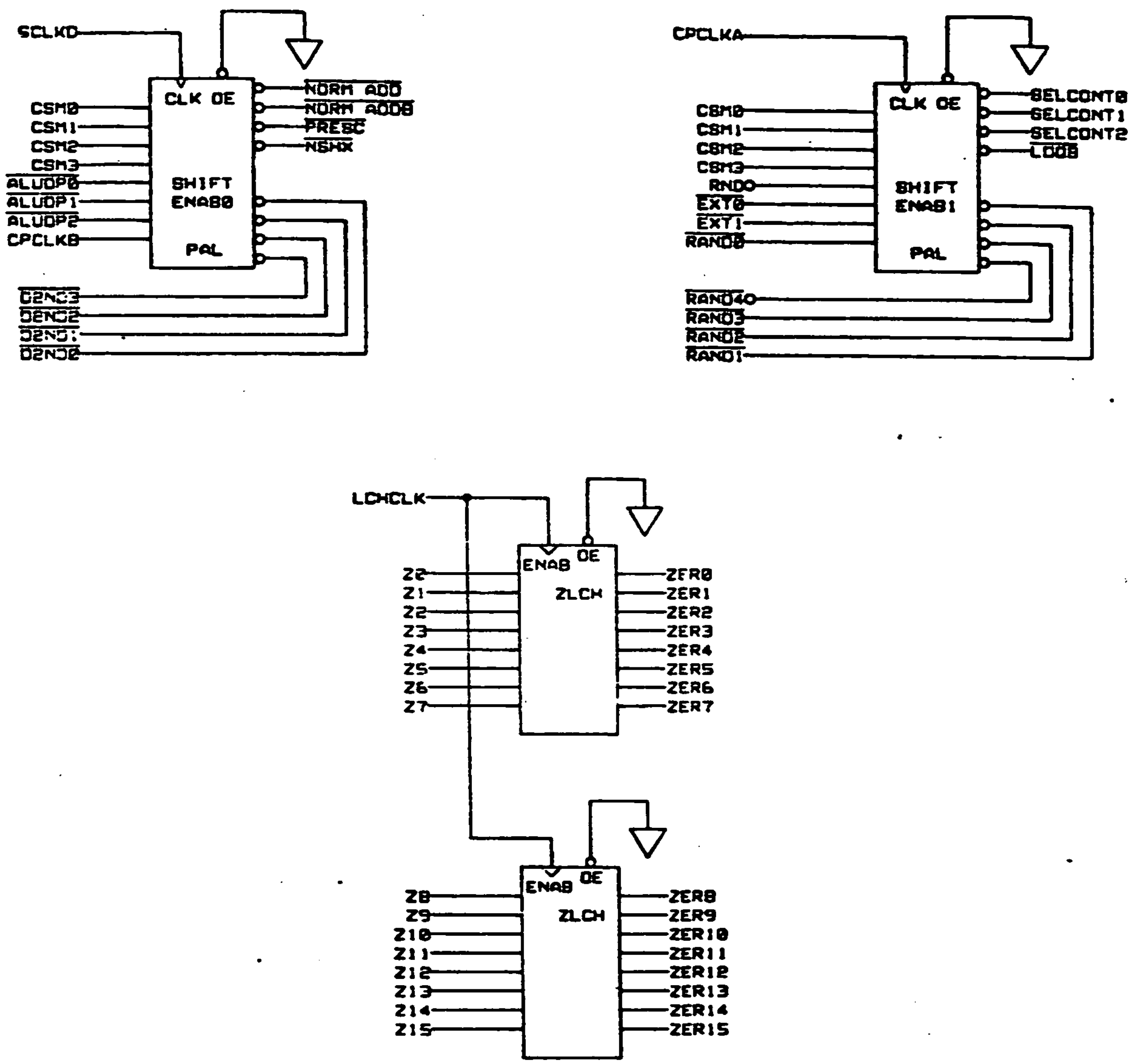

SCLKD
CLK OE
CSM0
CSM1
CSM2
CSM3
ALUOP0
ALUOP1
ALUOP2
CPCLKB
SHIFT ENAB0
PAL
NORM ADD
NORM ADDB
PRESC
NSHX
D2NC3
D2NC2
D2NC1
D2NC0
CPCLKA
CLK OE
CSM0
CSM1
CSM2
CSM3
RNDO
EXT0
EXT1
RAND0
SHIFT ENAB1
PAL
SELCONT0
SELCONT1
SELCONT2
LODB
RAND4
RAND3
RAND2
RAND1
LCHCLK
ENAB OE
ZLCH
Z0
Z1
Z2
Z3
Z4
Z5
Z6
Z7
ZER0
ZER1
ZER2
ZER3
ZER4
ZER5
ZER6
ZER7
ENAB OE
ZLCH
Z8
Z9
Z10
Z11
Z12
Z13
Z14
Z15
ZER8
ZER9
ZER10
ZER11
ZER12
ZER13
ZER14
ZER15

FIG. 26

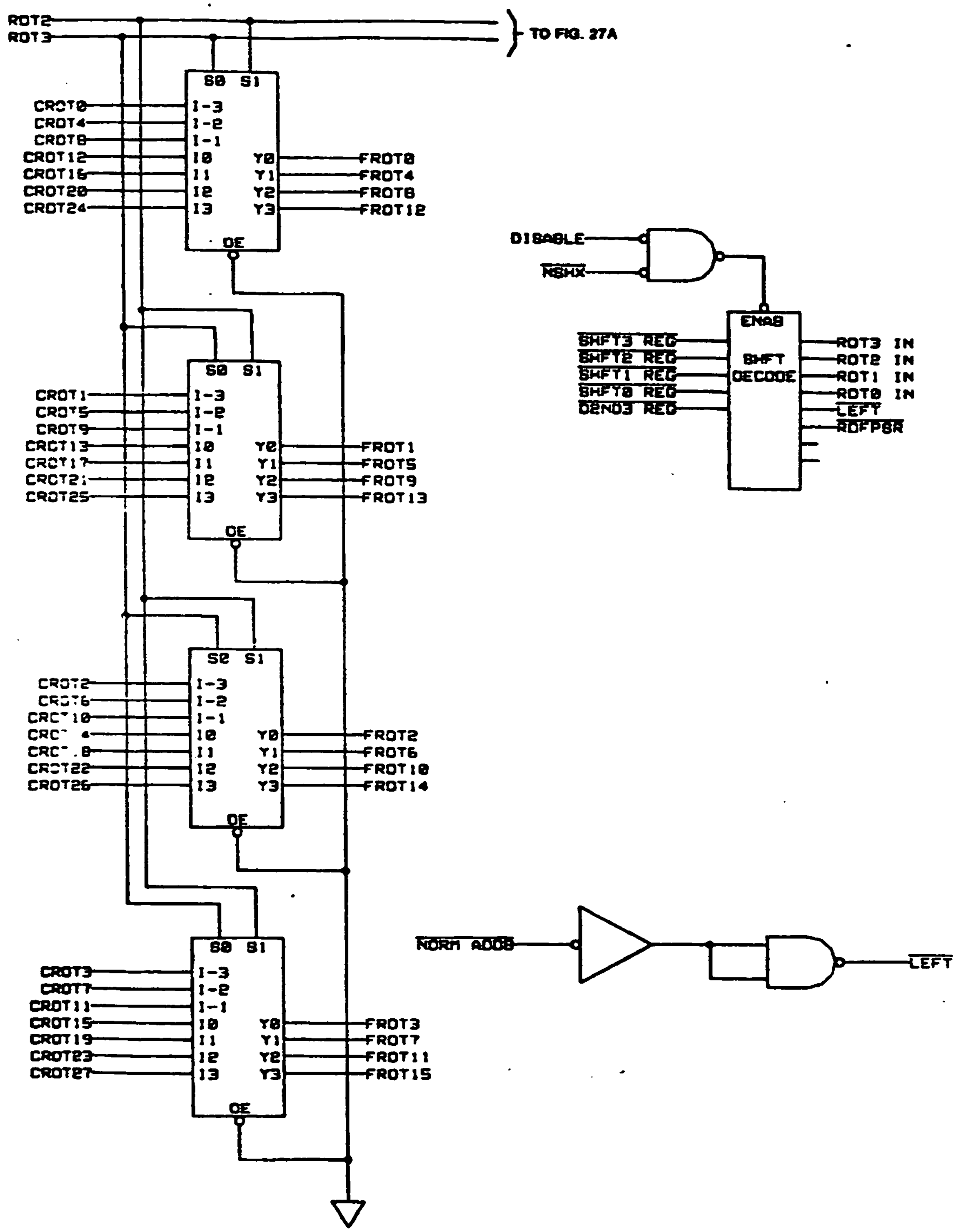

ROT2
ROT3
TO FIG. 27A
S0 S1
CROT0 CROT4 CROT8 CROT12 CROT16 CROT20 CROT24
I-3 I-2 I-1 I0 I1 I2 I3
Y0 Y1 Y2 Y3
FROT0 FROT4 FROT8 FROT12
OE
CROT1 CROT5 CROT9 CROT13 CROT17 CROT21 CROT25
FROT1 FROT5 FROT9 FROT13
CROT2 CROT6 CROT10 CROT14 CROT18 CROT22 CROT26
FROT2 FROT6 FROT10 FROT14
CROT3 CROT7 CROT11 CROT15 CROT19 CROT23 CROT27
FROT3 FROT7 FROT11 FROT15
DISABLE
NSHX
ENAB
SHFT DECODE
SHFT3 REG
SHFT2 REG
SHFT1 REG
SHFT0 REG
D2N03 REG
ROT3 IN
ROT2 IN
ROT1 IN
ROT0 IN
LEFT
RDFPSR
NORM ADDB
LEFT

FIG. 27

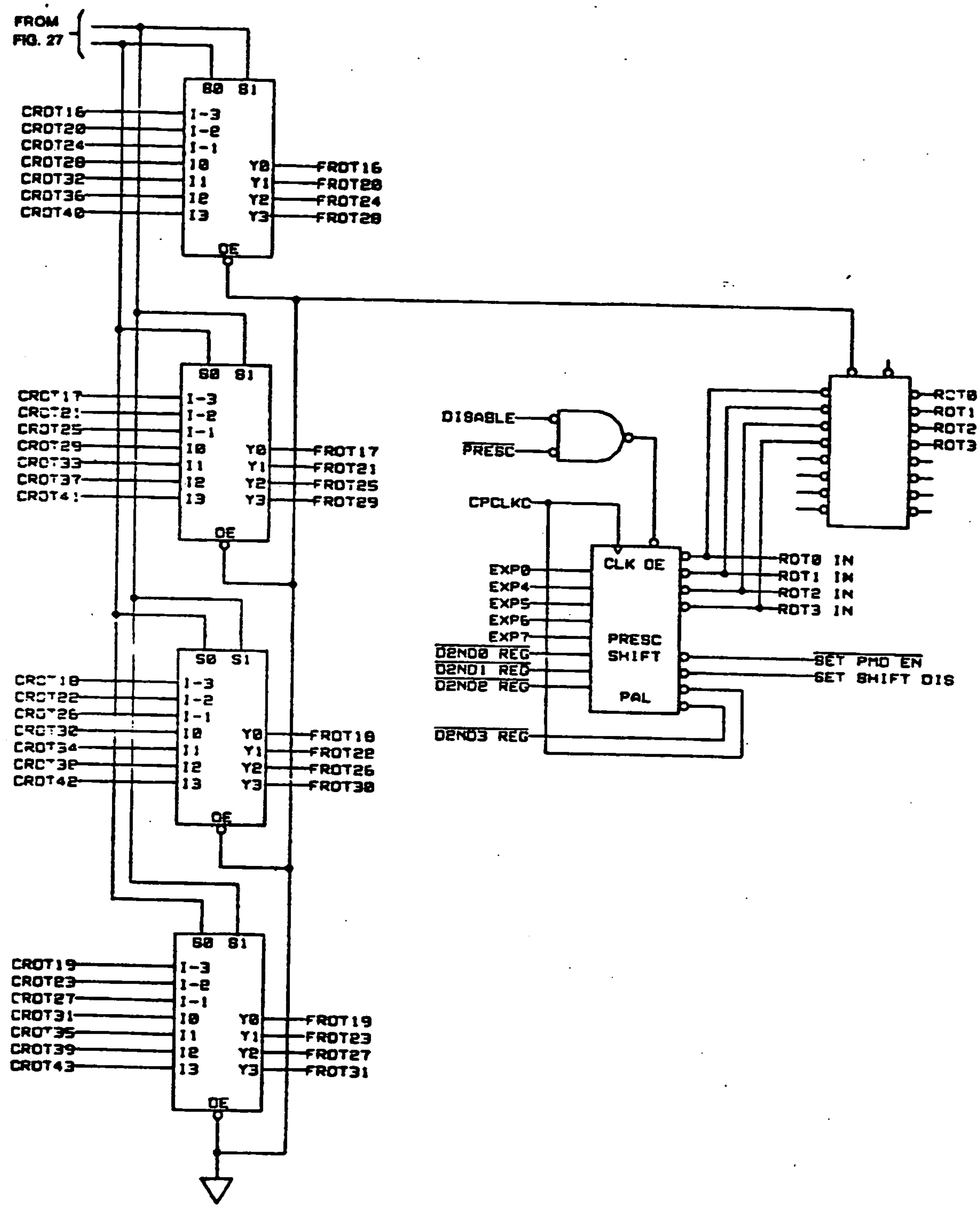

FROM FIG. 27
S0 S1
CROT16
CROT20
CROT24
CROT28
CROT32
CROT36
CROT40
I-3
I-2
I-1
I0
I1
I2
I3
Y0
Y1
Y2
Y3
OE
FROT16
FROT20
FROT24
FROT28
CROT17
CROT21
CROT25
CROT29
CROT33
CROT37
CROT41
FROT17
FROT21
FROT25
FROT29
CROT18
CROT22
CROT26
CROT30
CROT34
CROT38
CROT42
FROT18
FROT22
FROT26
FROT30
CROT19
CROT23
CROT27
CROT31
CROT35
CROT39
CROT43
FROT19
FROT23
FROT27
FROT31
DISABLE
PRESC
CPCLKC
EXP0
EXP4
EXP5
EXP6
EXP7
D2ND0 REG
D2ND1 REG
D2ND2 REG
D2ND3 REG
CLK OE
PRESC
SHIFT
PAL
ROT0 IN
ROT1 IN
ROT2 IN
ROT3 IN
SET PMD EN
SET SHIFT DIS
ROT0
ROT1
ROT2
ROT3

FIG. 27A

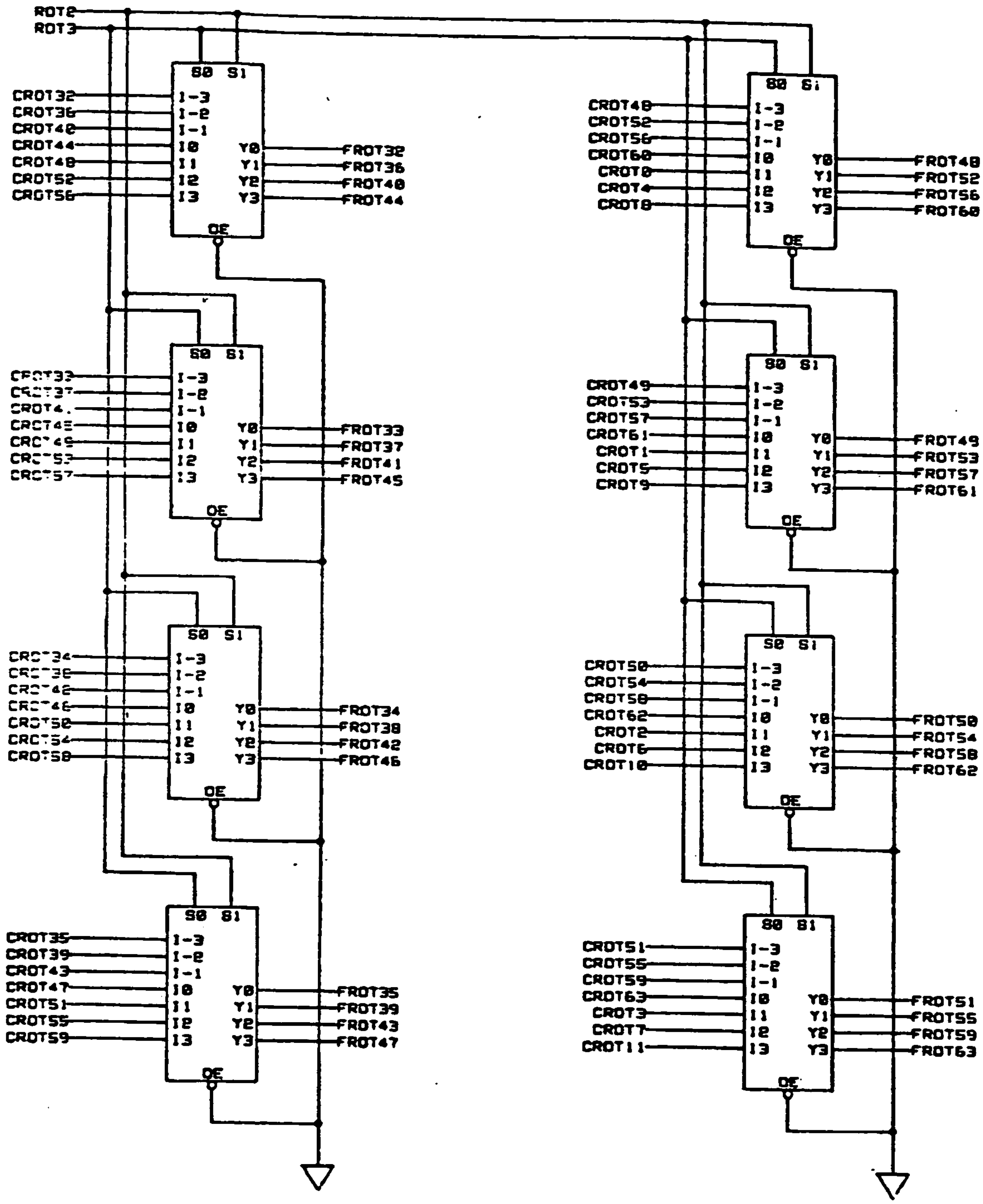

ROT2
ROT3
S0 S1
I-3
I-2
I-1
I0
I1
I2
I3
Y0
Y1
Y2
Y3
OE
CROT32
CROT36
CROT40
CROT44
CROT48
CROT52
CROT56
FROT32
FROT36
FROT40
FROT44
FROT33
FROT37
FROT41
FROT45
FROT34
FROT38
FROT42
FROT46
CROT58
CROT35
CROT39
CROT43
CROT47
CROT51
CROT55
CROT59
FROT35
FROT39
FROT43
FROT47
CROT48
CROT52
CROT56
CROT60
CROT0
CROT4
CROT8
FROT48
FROT52
FROT56
FROT60
CROT49
CROT53
CROT57
CROT61
CROT1
CROT5
CROT9
FROT49
FROT53
FROT57
FROT61
CROT50
CROT54
CROT58
CROT62
CROT2
CROT6
CROT10
FROT50
FROT54
FROT58
FROT62
CROT51
CROT55
CROT59
CROT63
CROT3
CROT7
CROT11
FROT51
FROT55
FROT59
FROT63

FIG. 28

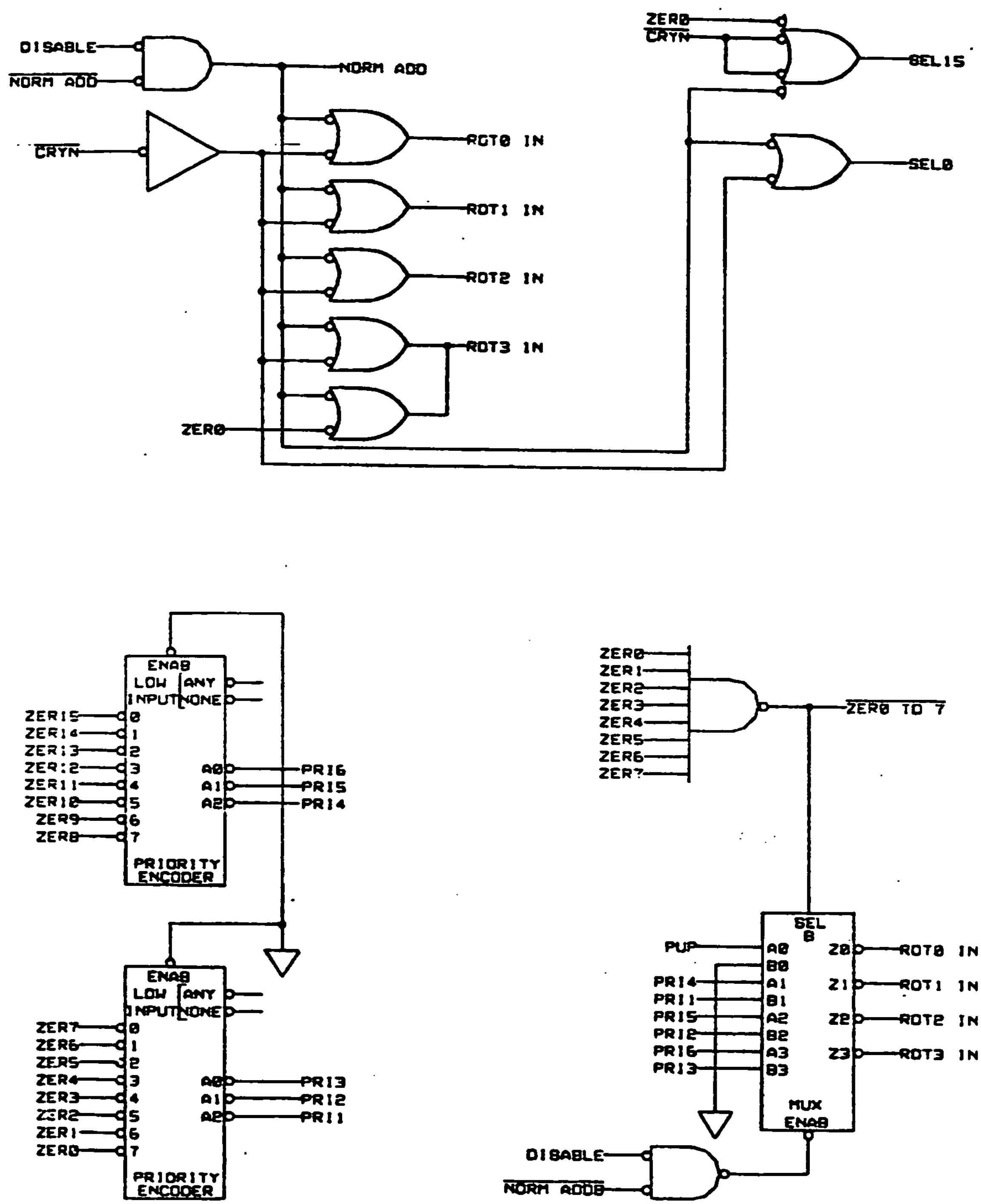

DISABLE
NORM ADD
NORM ADD
CRYN
ROT0 IN
ROT1 IN
ROT2 IN
ROT3 IN
ZERO
ZERO
CRYN
SEL15
SEL0
ENAB
LOW INPUT
ANY
NONE
ZER15
ZER14
ZER13
ZER12
ZER11
ZER10
ZER9
ZER8
A0
A1
A2
PR16
PR15
PR14
PRIORITY ENCODER
ZER7
ZER6
ZER5
ZER4
ZER3
ZER2
ZER1
ZER0
PR13
PR12
PR11
ZERO TO 7
SEL B
PUP
MUX ENAB
Z0
Z1
Z2
Z3

FIG. 29

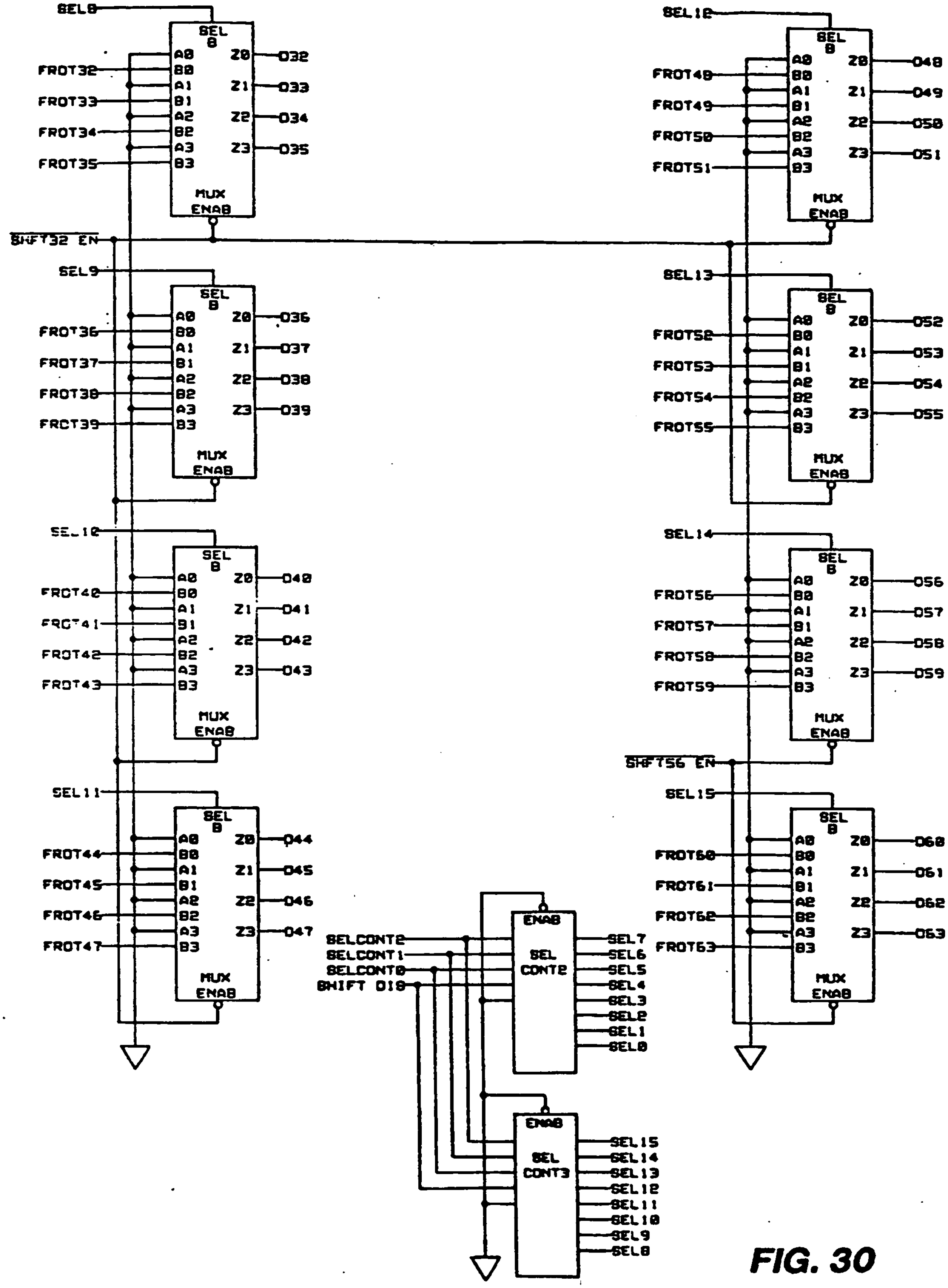

SEL8
SEL B
A0 B0 A1 B1 A2 B2 A3 B3
Z0 Z1 Z2 Z3
MUX ENAB
FROT32
FROT33
FROT34
FROT35
O32
O33
O34
O35
SHFT32 EN
SEL9
FROT36
FROT37
FROT38
FROT39
O36
O37
O38
O39
SEL10
FROT40
FROT41
FROT42
FROT43
O40
O41
O42
O43
SEL11
FROT44
FROT45
FROT46
FROT47
O44
O45
O46
O47
SEL12
FROT48
FROT49
FROT50
FROT51
O48
O49
O50
O51
SEL13
FROT52
FROT53
FROT54
FROT55
O52
O53
O54
O55
SEL14
FROT56
FROT57
FROT58
FROT59
O56
O57
O58
O59
SHFT56 EN
SEL15
FROT60
FROT61
FROT62
FROT63
O60
O61
O62
O63
SELCONT2
SELCONT1
SELCONT0
SHIFT O10
ENAB
SEL CONT2
SEL7
SEL6
SEL5
SEL4
SEL3
SEL2
SEL1
SEL0
SEL CONT3
SEL15
SEL14
SEL13
SEL12
SEL11
SEL10
SEL9
SEL8

FIG. 30

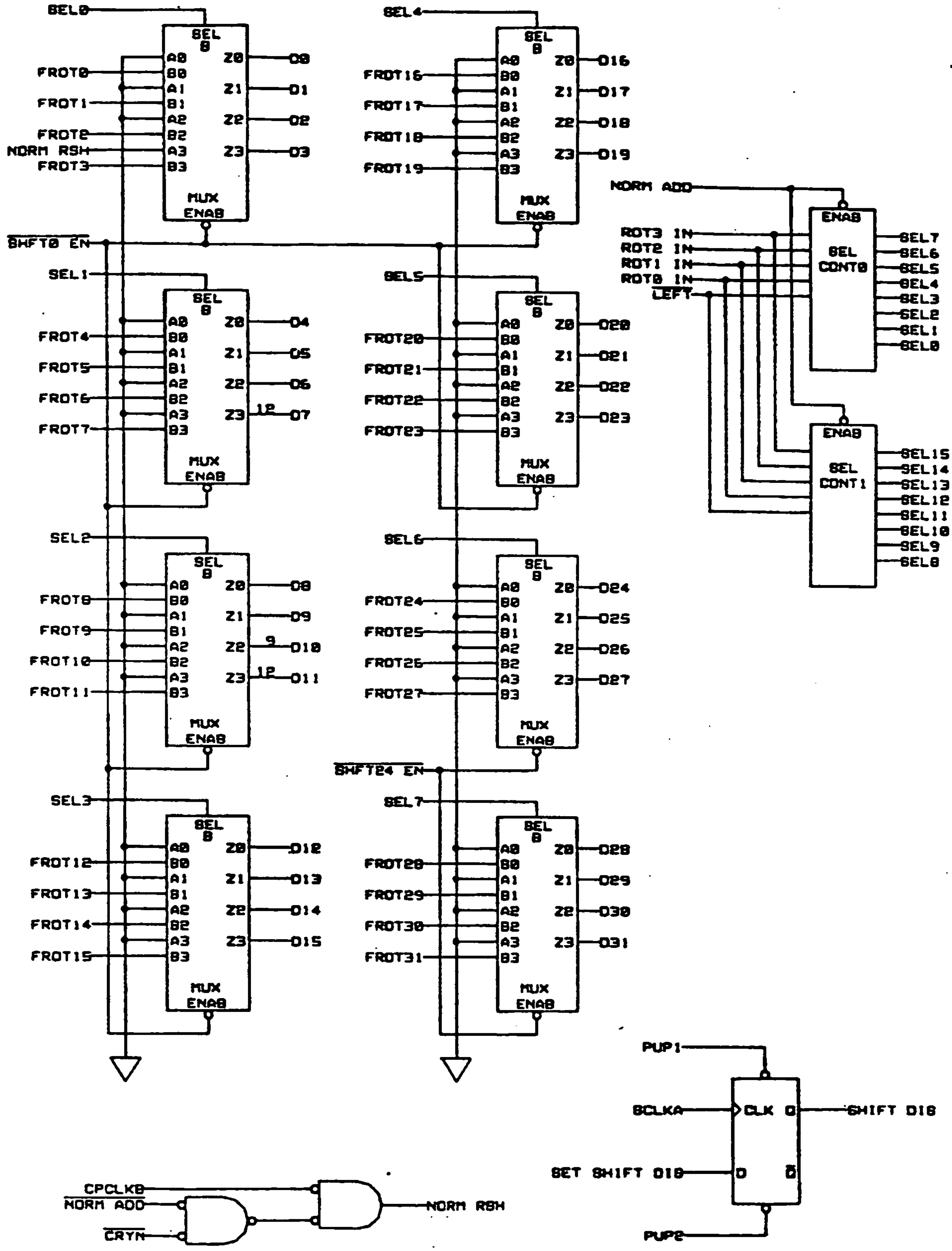

SEL0
FROT0
FROT1
FROT2
NORM RSH
FROT3
SHFT0 EN
SEL1
FROT4
FROT5
FROT6
FROT7
SEL2
FROT8
FROT9
FROT10
FROT11
SEL3
FROT12
FROT13
FROT14
FROT15
SEL4
FROT16
FROT17
FROT18
FROT19
SEL5
FROT20
FROT21
FROT22
FROT23
SEL6
FROT24
FROT25
FROT26
FROT27
SHFT24 EN
SEL7
FROT28
FROT29
FROT30
FROT31
SEL 8
MUX ENAB
A0 B0 A1 B1 A2 B2 A3 B3
Z0 Z1 Z2 Z3
D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 D10 D11 D12 D13 D14 D15
D16 D17 D18 D19 D20 D21 D22 D23 D24 D25 D26 D27 D28 D29 D30 D31
NORM ADD
ROT3 IN
ROT2 IN
ROT1 IN
ROT0 IN
LEFT
ENAB
SEL CONT0
SEL CONT1
SEL7 SEL6 SEL5 SEL4 SEL3 SEL2 SEL1 SEL0
SEL15 SEL14 SEL13 SEL12 SEL11 SEL10 SEL9 SEL8
PUP1
BCLKA
CLK
SET SHIFT D18
SHIFT D18
PUP2
CPCLKB
NORM ADD
CRYN
NORM RSH

FIG. 31

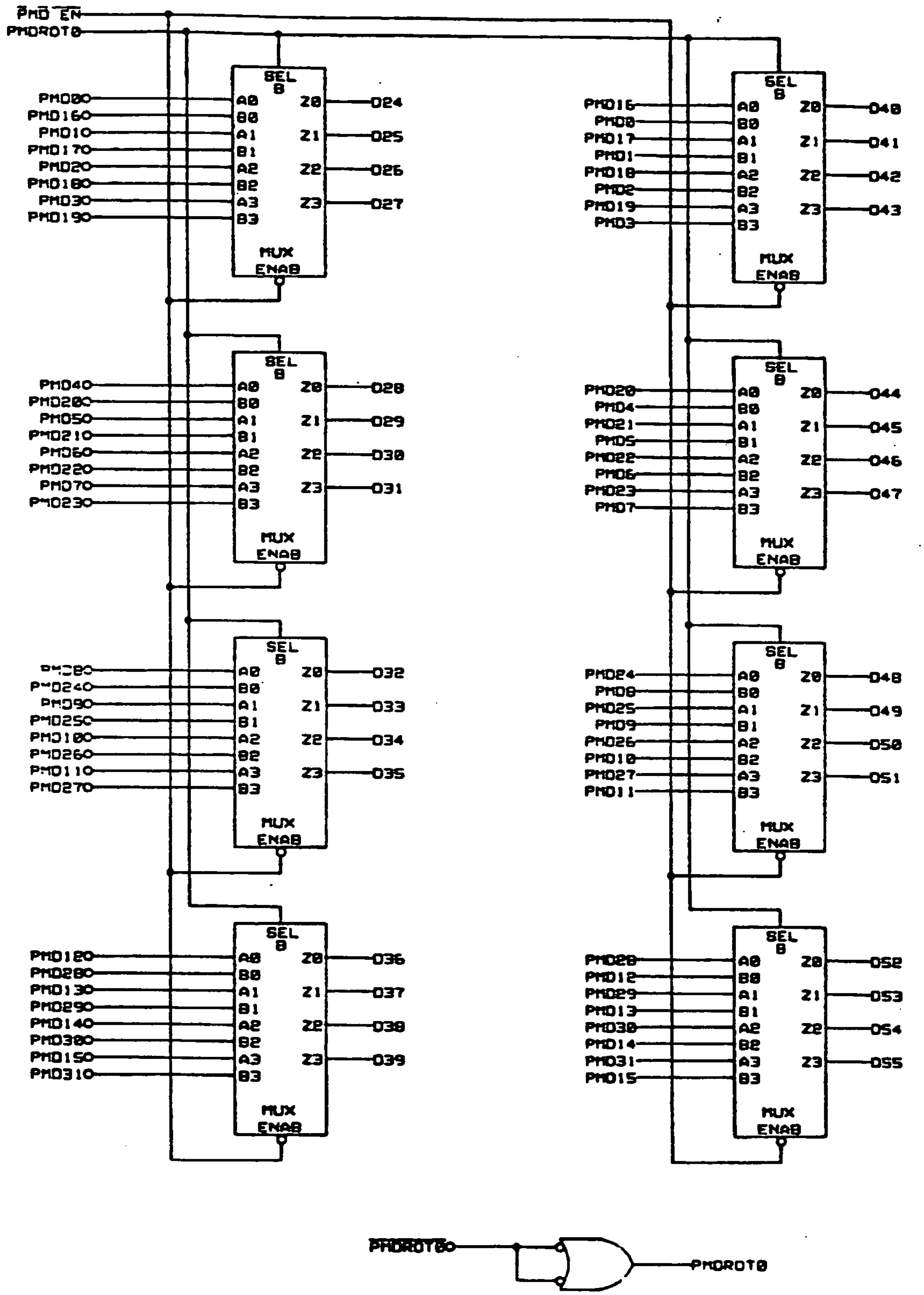

PMD EN
PMDROT0
SEL B
MUX ENAB
A0
B0
A1
B1
A2
B2
A3
B3
Z0
Z1
Z2
Z3
PMD00
PMD160
PMD10
PMD170
PMD20
PMD180
PMD30
PMD190
O24
O25
O26
O27
PMD40
PMD200
PMD50
PMD210
PMD60
PMD220
PMD70
PMD230
O28
O29
O30
O31
PMD240
PMD90
PMD250
PMD100
PMD260
PMD110
PMD270
O32
O33
O34
O35
PMD120
PMD280
PMD130
PMD290
PMD140
PMD300
PMD150
PMD310
O36
O37
O38
O39
PMD16
PMD0
PMD17
PMD1
PMD18
PMD2
PMD19
PMD3
O40
O41
O42
O43
PMD20
PMD4
PMD21
PMD5
PMD22
PMD6
PMD23
PMD7
O44
O45
O46
O47
PMD24
PMD8
PMD25
PMD9
PMD26
PMD10
PMD27
PMD11
O48
O49
O50
O51
PMD28
PMD12
PMD29
PMD13
PMD30
PMD14
PMD31
PMD15
O52
O53
O54
O55
PMDROT0
PMDROT0

FIG. 32

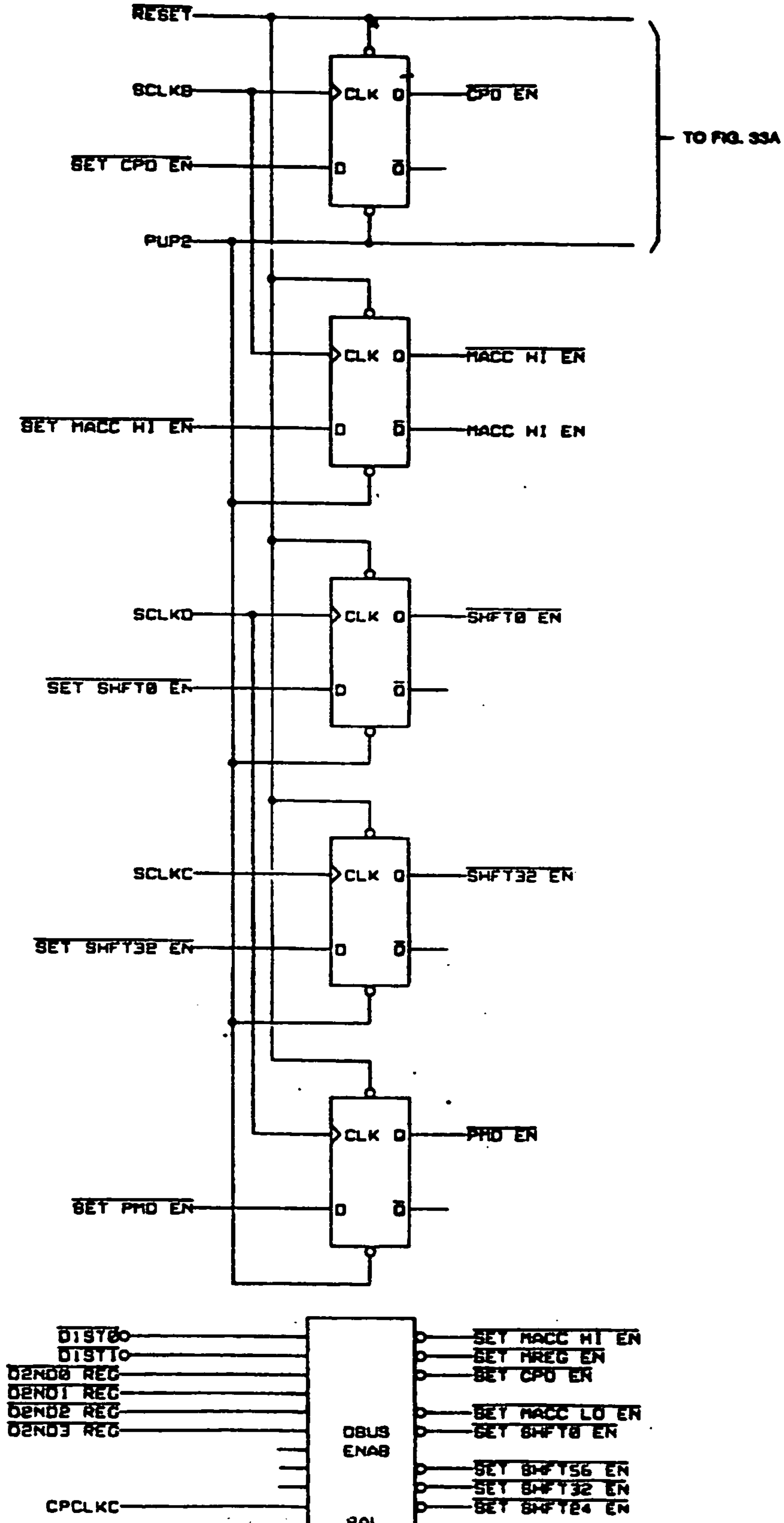
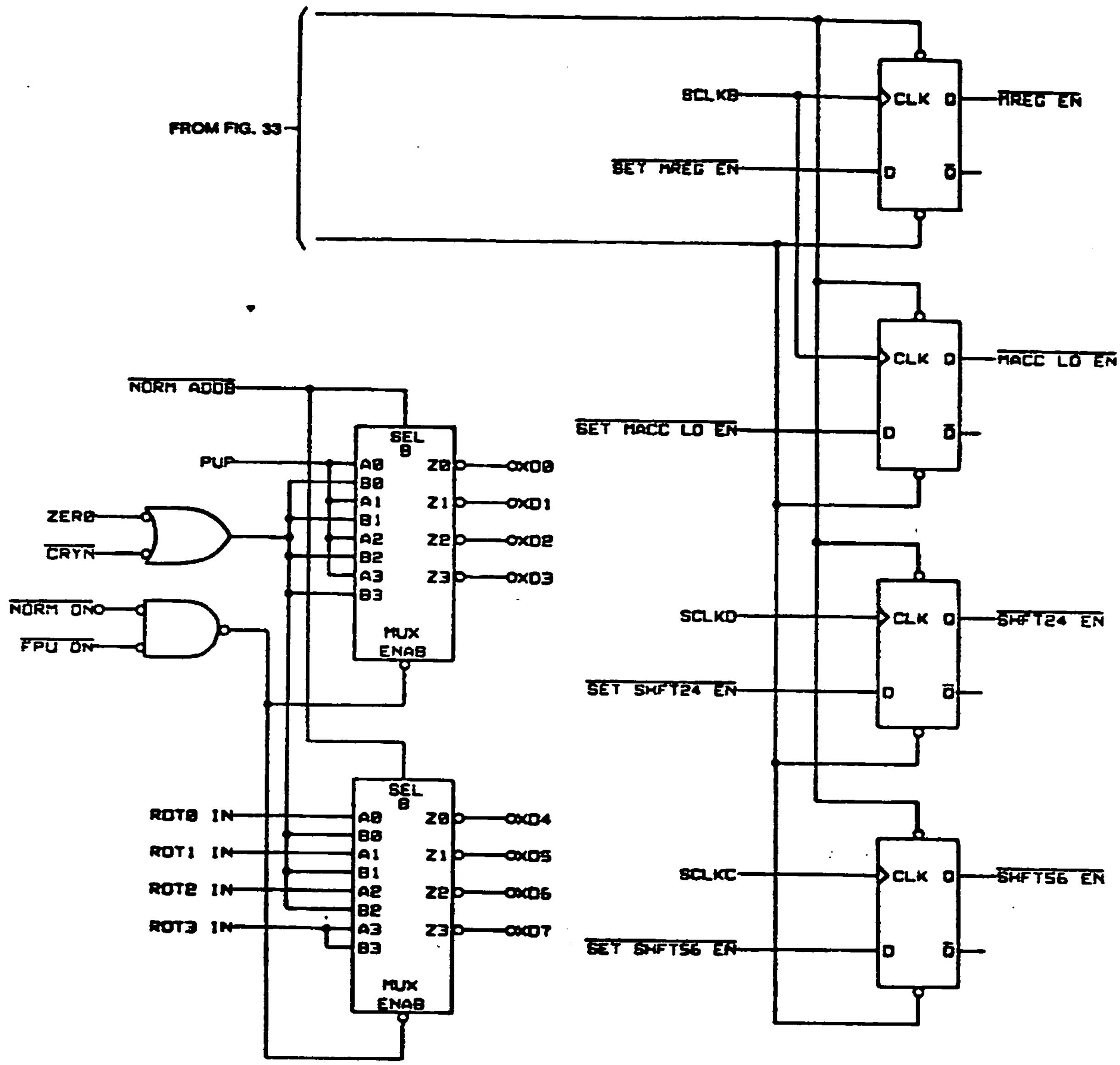

RESET
SCLKB
CLK
Q
CPO EN
SET CPO EN
D
Q
TO FIG. 33A
PUP2
MACC HI EN
SET MACC HI EN
MACC HI EN
SCLKD
SHFT8 EN
SET SHFT8 EN
SCLKC
SHFT32 EN
SET SHFT32 EN
PMD EN
SET PMD EN
DIST0
DIST1
D2ND0 REC
D2ND1 REC
D2ND2 REC
D2ND3 REC
CPCLKC
DBUS
ENAB
PAL
SET MACC HI EN
SET MREG EN
SET CPO EN
SET MACC LO EN
SET SHFT8 EN
SET SHFT56 EN
SET SHFT32 EN
SET SHFT24 EN

FIG. 33



FROM FIG. 33
SCLKB
SET MREG EN
CLK
D
Q
MREG EN
MACC LO EN
SET MACC LO EN
SCLKD
SET SHFT24 EN
SHFT24 EN
SCLKC
SET SHFT56 EN
SHFT56 EN
NORM ADDB
PUP
ZERO
CRYN
NORM ON
FPU ON
SEL B
A0 B0 A1 B1 A2 B2 A3 B3
Z0 Z1 Z2 Z3
MUX ENAB
OXD0
OXD1
OXD2
OXD3
ROT0 IN
ROT1 IN
ROT2 IN
ROT3 IN
OXD4
OXD5
OXD6
OXD7

FIG. 33A

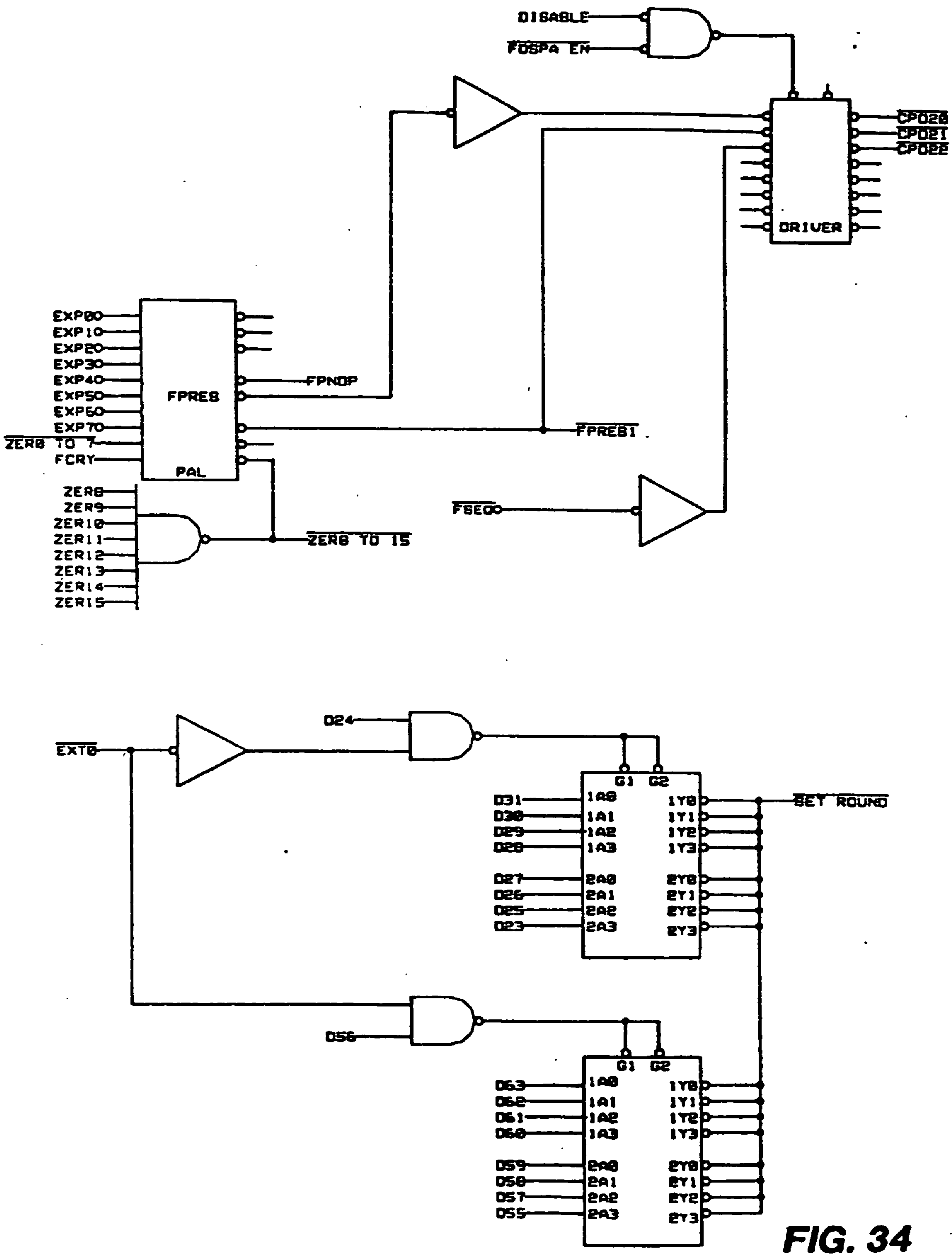

DISABLE
FOSPA EN
CPO20
CPO21
CPO22
DRIVER
EXP00
EXP10
EXP20
EXP30
EXP40
EXP50
EXP60
EXP70
ZER0 TO 7
FCRY
FPREB
PAL
FPNOP
FPREB1
ZER8
ZER9
ZER10
ZER11
ZER12
ZER13
ZER14
ZER15
ZER8 TO 15
FSEQ
EXT0
D24
G1 G2
D31
D30
D29
D28
D27
D26
D25
D23
1A0
1A1
1A2
1A3
2A0
2A1
2A2
2A3
1Y0
1Y1
1Y2
1Y3
2Y0
2Y1
2Y2
2Y3
SET ROUND
D56
D63
D62
D61
D60
D59
D58
D57
D55

FIG. 34

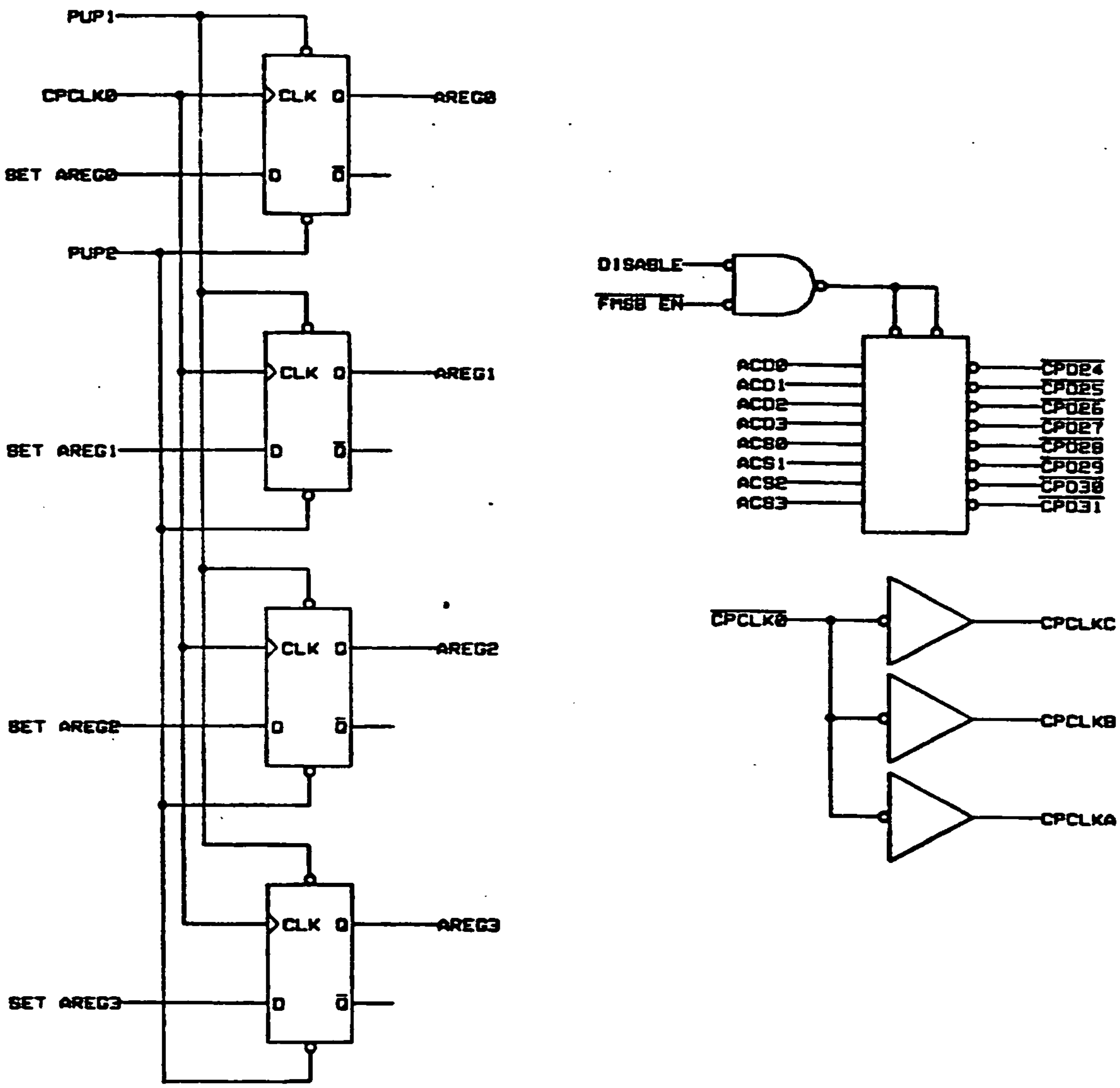

PUP1
CPCLK0
CLK
Q
AREG0
SET AREG0
D
Q̄
PUP2
AREG1
SET AREG1
AREG2
SET AREG2
AREG3
SET AREG3
DISABLE
FMSB EN
ACD0
ACD1
ACD2
ACD3
ACS0
ACS1
ACS2
ACS3
CPD24
CPD25
CPD26
CPD27
CPD28
CPD29
CPD30
CPD31
CPCLK0
CPCLKC
CPCLKB
CPCLKA

FIG. 35

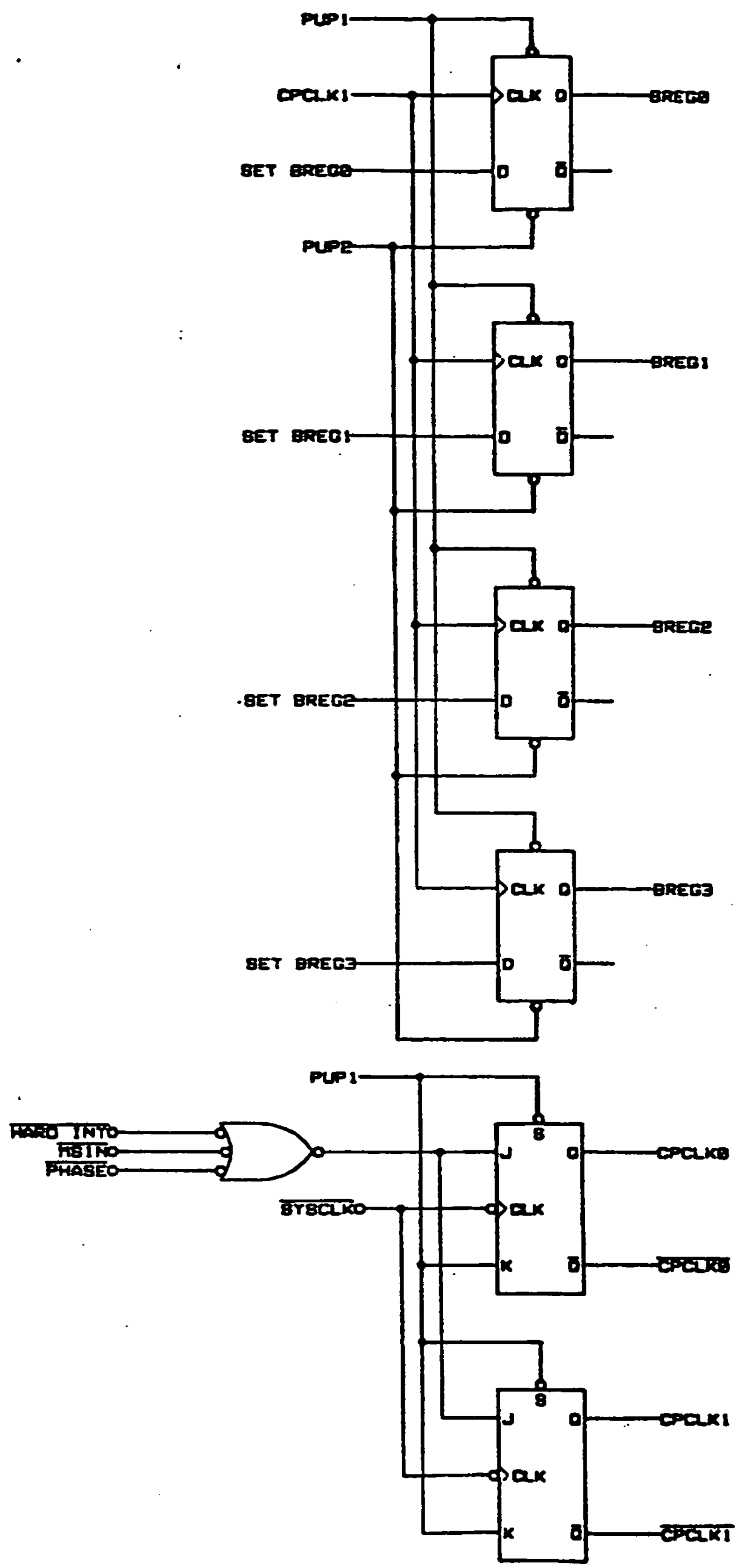

PUP1
CPCLK1
SET BREG0
PUP2
SET BREG1
SET BREG2
SET BREG3
CLK
D
Q
BREG0
BREG1
BREG2
BREG3
PUP1
HARD INT0
RSIN0
PHASE0
SYSCLK0
S
J
K
CPCLK0
CPCLK1

FIG. 35A

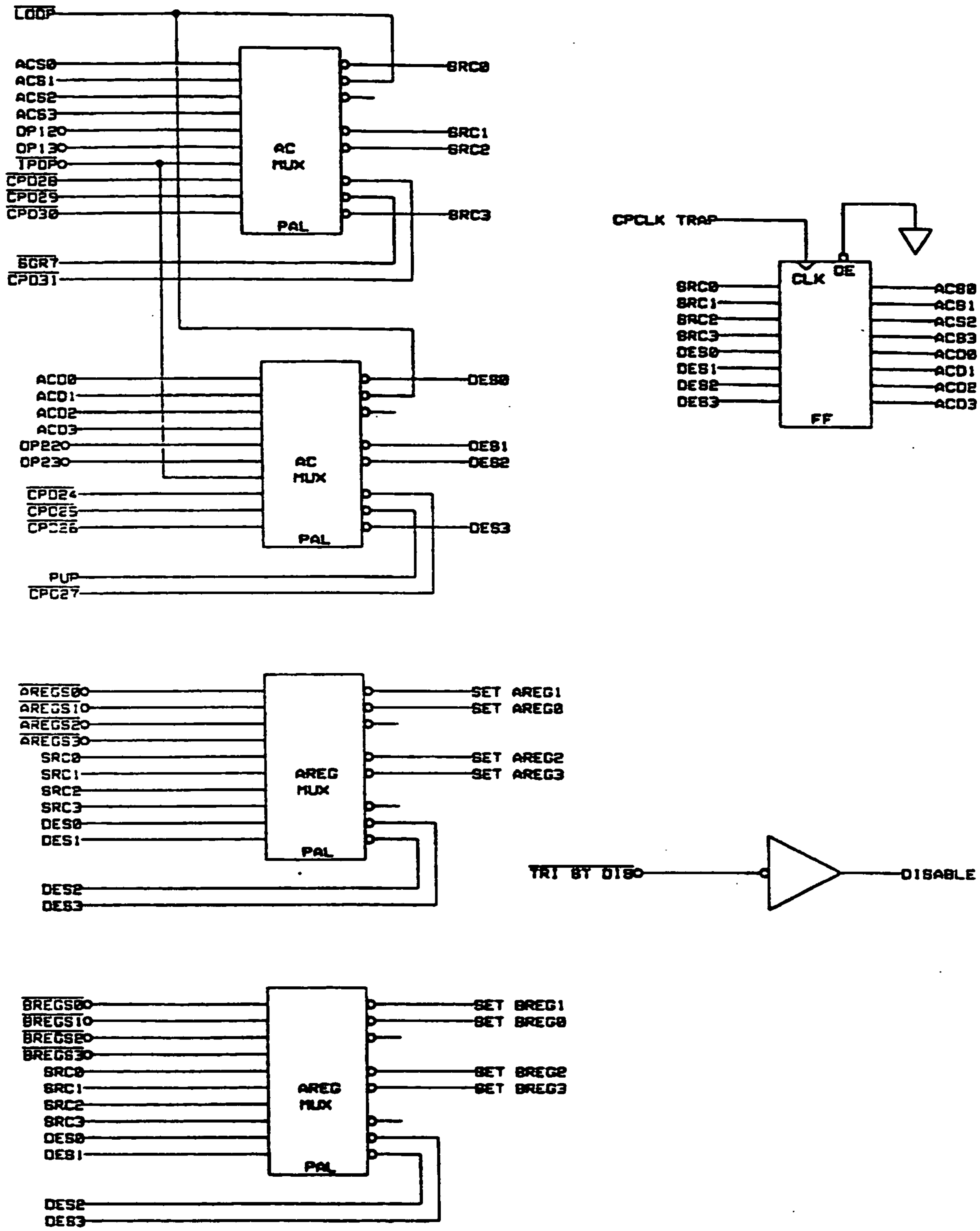

LOOP
ACS0
ACS1
ACS2
ACS3
OP120
OP130
IPDP
CPD28
CPD29
CPD30
SGR7
CPD31
AC MUX
PAL
SRC0
SRC1
SRC2
SRC3
ACD0
ACD1
ACD2
ACD3
OP220
OP230
CPD24
CPD25
CPD26
PUP
CPD27
AC MUX
PAL
DES0
DES1
DES2
DES3
CPCLK TRAP
CLK
OE
SRC0
SRC1
SRC2
SRC3
DES0
DES1
DES2
DES3
FF
ACS0
ACS1
ACS2
ACS3
ACD0
ACD1
ACD2
ACD3
AREGS0
AREGS1
AREGS2
AREGS3
SRC0
SRC1
SRC2
SRC3
DES0
DES1
DES2
DES3
AREG MUX
PAL
SET AREG1
SET AREG0
SET AREG2
SET AREG3
TRI ST DIS
DISABLE
BREGS0
BREGS1
BREGS2
BREGS3
SRC0
SRC1
SRC2
SRC3
DES0
DES1
DES2
DES3
AREG MUX
PAL
SET BREG1
SET BREG0
SET BREG2
SET BREG3

FIG. 36

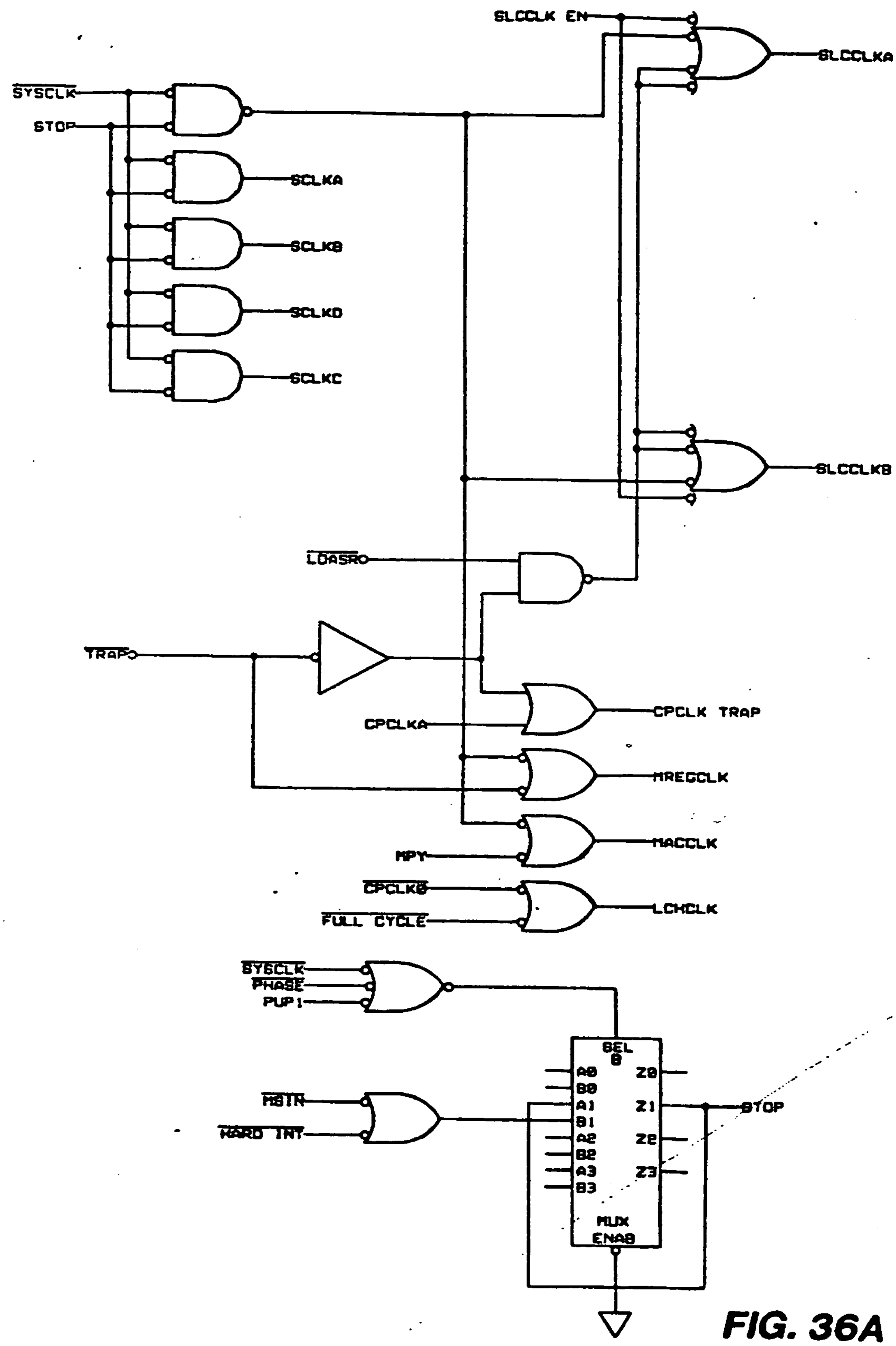

SLCCLK EN
SLCCLKA
SYSCLK
STOP
SCLKA
SCLKB
SCLKD
SCLKC
SLCCLKB
LDASRO
TRAP
CPCLK TRAP
CPCLKA
MREGCLK
MACCLK
MPY
CPCLKB
LCHCLK
FULL CYCLE
SYSCLK
PHASE
PUP1
SEL
B
A0
B0
A1
B1
A2
B2
A3
B3
Z0
Z1
Z2
Z3
STOP
MSIN
HARD INT
MUX
ENAB

FIG. 36A